Ontology Verbalization using Semantic-Refinement

Vinu E.V * and P Sreenivasa Kumar
Department of Computer Science and Engineering, Indian Institute of Technology Madras, Chennai, India
E-mail: {vinuev,psk}@cse.iitm.ac.in

Abstract. In this paper, we propose a rule-based technique to generate redundancy-free natural language (NL) descriptions of Web Ontology Language (OWL) entities. The existing approaches which address the problem of verbalizing OWL ontologies generate NL text segments which are close to their counterpart statements in the ontology. Some of these approaches also perform grouping and aggregating of these NL text segments to generate a more fluent and comprehensive form of the content. Restricting our attention to description of individuals and atomic concepts, we find that the approach currently followed in the available tools is that of determining the set of all logical conditions that are satisfied by the given individual/concept name and translate these conditions verbatim into corresponding NL descriptions. Human-understandability of such descriptions is affected by the presence of repetitions and redundancies, as they have high fidelity to the OWL representation of the entities. In the literature, no efforts had been taken to remove redundancies and repetitions at the logical level before generating the NL descriptions of entities and we find this to be the main reason for lack of readability of the generated text. In this paper, we propose a technique called semantic-refinement to generate meaningful and easily-understandable (what we call redundancy-free) text descriptions of individuals and concepts of a given OWL ontology. We identify the combinations of OWL/DL constructs that lead to repetitive/redundant descriptions and propose a series of refinement rules to rewrite the conditions that are satisfied by an individual/concept in a meaning-preserving manner. The reduced set of conditions are then employed for generating textual descriptions. Our experiments show that, semantic-refinement technique leads to significantly improved descriptions of ontology entities. We also test the effectiveness and usefulness of the generated descriptions for the purpose of validating the ontologies and find that the proposed technique is indeed helpful in the context. The details of an empirical study to support the claim are provided in the paper.

Keywords: Verbalization, Ontologies, Rule-based system

1. Introduction

Web Ontology Language (OWL/DL) ontologies are knowledge representation structures which are based on decidable fragments of first order logic. They model domain knowledge in the form of logical axioms; so that, an intelligent agent with the help of a reasoning system, can make use of them for several applications. Ontologies play an important role in the development and deployment of the Semantic Web since they help in enhancing the understanding of the contextual meaning of data. Since the knowledge in the form of an ontology is inherently characterized by complex relational contexts, it is typically inaccessible for non-Semantic Web experts. This problem motivated researchers to work on natural language (NL) verbalization techniques for OWL ontologies. The existing approaches in this direction mainly strive for one-to-one conversion of logical statements to NL texts, and re-
result in methods which produce verbatim equivalents of OWL constructs. One of the main and common drawbacks of these approaches is that, since the generated sentences are verbatim equivalent to the OWL statements, they are likely to have high amount of redundancy. As we show later with examples, it can be very annoying for a human reader to read and understand such sentences. Therefore, in this paper, we explore techniques which can generate NL sentences that do not have redundancies and are semantically equivalent to their OWL counterparts.

We will closely look at the problem of verbalization of OWL ontologies from the perspective of using the generated descriptions for validating the formalized knowledge. Typically, ontologies are developed by a group of knowledge engineers with the help of domain experts. The domain experts provide the knowledge to be formalized and the engineers build the ontology out of it. Since an ontology development involves multiple parties (engineers and domain experts), the process usually follows a Spiral model, where suitable feedback mechanisms are involved to improve the structure.

As an ontology evolves over a period of time, it can grow in size and complexity. Unless the updates are carefully carried out, the quality of the ontology might degrade. To prevent such quality depletion, usually an ontology development cycle is accompanied by a validation phase, where both the knowledge engineers and domain experts meet to review the content of the ontology.

In a typical validation phase, new axioms are included or existing axioms are altered or removed, to maintain the correctness of the ontology. The conventional method for incorporating new axioms and validating the ontology involves a validity check by domain experts. Domain experts, who do the validity check, cannot be expected to be highly knowledgeable on formal methods and notations. For their convenience, the OWL axioms will have to be first converted into corresponding NL texts. Ontology verbalizers and ontology authoring tools such as ACE [9], NaturalOWL [1] and SWAT Tools [12], can be utilized for generating controlled natural language (CNL) descriptions of OWL statements. Restricting our attention to description of individuals and atomic concepts, we find that the approach currently followed in the available tools is that of determining the set of all logical conditions that are satisfied by the given individual/concept name and translate these conditions verbatim into corresponding NL descriptions. But the verbatim fidelity of such descriptions to the underlying OWL statements, makes them a poor choice for ontology validation. This is because, the descriptions will be confusing to a person who is not familiar with formal constructs, and it will be difficult to correctly understand the meaning from such descriptions. This issue had been previously reported in papers such as [11,12], where the authors tried to overcome the issue by applying operations such as grouping and aggregation on the verbalized text. But, since the issue had been treated at the NL text level, the opportunity for a logical-level refinement of the OWL statements to generate a more meaningful and human-understandable representation has been ignored.

For example, consider the following logical axioms (from People & Pets ontology)\(^1\) represented in the description logic (DL) notation.

1. \(\text{Cat} \sqsubseteq \text{Person} \sqcap \exists \text{hasPet.Animal} \sqcap \exists \text{hasPet.Cat}\)
2. \(\text{Cat} \text{Owner}(\text{sam})\)
3. \(\text{Cat} \sqsubseteq \text{Animal}\)

The different variants of the CNL sentences correspond to the individual \(\text{sam}\) as follows:

- A cat-owner is a person. A cat-owner has as pet an animal. A cat-owner has as pet a cat. \(\text{Sam} \text{ is a cat-owner. All cats are animals.}\)  
or (with grouping and aggregation)  
- A cat-owner is a person. A cat-owner is all of the following: something that has pet an animal, and something that has pet a cat. Example: \(\text{sam. All cats are animals.}\)

As can be easily seen, these descriptions have redundant information and attempting verbatim equivalence to DL constructs has resulted in this situation. The above example illustrates one type of redundancy and several more are identified in the paper later.

In this paper, we introduce an approach for removing redundancies from the verbalized definitions of OWL/DL entities, and to generate the so-called redundancy-free representations/descriptions. We propose a technique called semantic-level refinement (or simply semantic-refinement) that helps in removing the redundant (portion of the) restrictions and generating a more semantically comprehensive description of the entity. From an application point of view, in this paper, we particularly focus on generating NL descrip-
tions of individuals and concepts for validating ontolo-
gies which follow SHIQ description logic.

Our proposed approach generates NL descriptions
of individuals and concepts by giving importance to
the semantic conciseness of the content. If we revis-
its and bidirectional mapping are helpful in enabling
the resulting NL expressions are unambiguous, they generate a collection of unordered
sentences that are difficult to comprehend.

To use these CNLs as a means for ontology author-
ing and for knowledge validation purposes, appropri-
ate organization of the verbalized text is necessary. A
detailed comparison of the systems that comprehend
the NL texts is given in [11]. Among such systems,
SWAT tools[3] are one of the recent and prominent tools
which use standard techniques from computational lin-
guistics to make the verbalized text more readable.
They tried to give better clarity to the generated text by
grouping, aggregation and elision. The Semantic Web
Authoring (SWAT) NL verbalization tools have given
much importance to the fluency of the verbalized sen-
tences [12], rather than removing redundancies from
their logical forms, hence have deficiencies in inter-
preting the ontology contents.

3. Preliminaries

3.1. SHIQ Ontologies

The description logic (DL) SHIQ is based on an
extension of the well-known logic ALC [10], with
added support for role hierarchies, inverse roles, tran-
sitive roles, and qualifying number restrictions [6].

We assume $N_C$ and $N_R$ as countably infinite dis-
joint sets of atomic concepts and atomic roles re-
spectively. A SHIQ role is either $R \in N_R$ or an inverse
role $R^\sim$ with $R \in N_R$. To avoid considering roles such
as $(R^\sim)^\sim$, we define a function $\text{Inv}(\cdot)$ which returns
the inverse of a role: $\text{Inv}(R) = R^\sim$ and $\text{Inv}(R^\sim) = R$.

The set of concepts in SHIQ is recursively defined
using the constructors in Table 1 where $A \in N_C, C, D$ are
concepts, $R, S$ are roles, and $n, m$ are positive in-
tegers. A SHIQ based ontology — denoted as a pair
$O = (T, A)$, where $T$ denotes terminological axioms
(also known as TBox) and $A$ represents assertional ax-
ioms (also known as ABox) — is a set of axioms of
the type specified in Table 1. A role $R$ in $O$ is transi-
tive if $\text{Tran}(R) \in O$ or $\text{Tran}(R^\sim) \in O$. Given an O,
let $\subseteq O$ be the smallest transitive reflexive relation
between roles $R_1$ and $R_2$, such that $R_1 \subseteq \subseteq R_2 \subseteq \subseteq O$ implies $R_1 \subseteq \subseteq R_2$ and $R_2 \subseteq \subseteq R_1$. For a SHIQ on-
tology $O$, the role $S$ in every concept of the form $\geq nS.C$
and $\leq mS.C$ in $O$, should be simple, that is, $R \subseteq \subseteq O$ holds
for no transitive role $R$ [2].

The semantics of SHIQ is defined using inter-
pretations. An interpretation is a pair $I = (\Delta_I, I^I)$ where
$\Delta_I$ is a non-empty set called the domain of the in-

\[ \text{Great Britain’s national mapping agency} \]

\[ \text{http://mcs.open.ac.uk/nlg/SWAT/} \]
terpretation and \( \mathcal{I} \) is the interpretation function. The function \( \mathcal{I} \) assigns a set \( A^X \subseteq \Delta \) to every \( A \in N_C \), and assigns a relation \( r^X \subseteq \Delta \times \Delta \) to every \( r \in N_R \).

The interpretation of the inverse role \( r^- \) is \( (r^-)^X := \{(y, x) \mid (x, y) \in r^X\} \). The interpretation is extended to concepts and axioms according to the rightmost column of Table 1 and Table 2 respectively, where \( \#X \) denotes the cardinality of the set \( X \).

We write \( \mathcal{I} \models \alpha \), if the interpretation \( \mathcal{I} \) satisfies the axiom \( \alpha \) (or \( \alpha \) is true in \( \mathcal{I} \)). \( \mathcal{I} \) is a model of an ontology \( \mathcal{O} \) (written \( \mathcal{I} \models \mathcal{O} \)) if \( \mathcal{I} \) satisfies every axiom in \( \mathcal{O} \). If we say \( \alpha \) is entailed by \( \mathcal{O} \), or \( \alpha \) is a logical consequence of \( \mathcal{O} \) (written \( \mathcal{O} \models \alpha \)), then every model of \( \mathcal{O} \) satisfies \( \alpha \). A concept \( C \) is subsumed by \( D \) w.r.t. \( \mathcal{O} \) if \( \mathcal{O} \models C \subseteq D \), and \( C \) is unsatisfiable w.r.t. \( \mathcal{O} \) if \( \mathcal{O} \models C \subseteq \bot \). Classification is the task of computing all subsumptions \( A \sqsubseteq B \) between atomic concepts such that \( A, B \in N_C \) and \( \mathcal{O} \models A \sqsubseteq B \); similarly, property classification of \( \mathcal{O} \) is the computation of all subsumptions between properties \( R \sqsubseteq S \) such that \( R, S \in N_R \) and \( \mathcal{O} \models R \sqsubseteq S \).

3.2. Running Example

In this section we introduce an example ontology (called the academic (ACAD) ontology) which we follow throughout this chapter. We have formalized various concepts in academic domain in this ontology. The ontology is rather small, but serves the purpose well. The TBox and ABox of the ontology is given in Table 3 and 4 respectively.

4. Newly Introduced Terminologies and Definitions

In this section, we introduce the terminologies and definitions by considering ontologies whose expressivity is bound to SHIQ description logic.

In this paper, we use the words “reduction” and “refinement” interchangeably. In the current context, ‘Description’ of an ontology entity refers to its domain-specific NL definition generated from the ontology.

4.1. Label-sets

To generate descriptions of individuals in an ontology, we associate with each individual a set of constraints it satisfies. We call these sets as label-sets in general. A Label-set of an individual is called a node-label-set and a label-set of a pair of individuals is called an edge-label-set. The rationale behind generating these label-sets is that, since all the constraints satisfied by an individual are captured at one place, it can easily be looked up for redundancies.

Node-label-set The node-label-set of an individual is the set which contains all the class expressions and (existential, universal and cardinality) restrictions satisfied by that individual.

**Definition 1** The node-label-set of an individual \( x \) (represented as \( L_\mathcal{O}(x) \)) is defined as:

\[ L_\mathcal{O}(x) = \{ c_i \mid \mathcal{O} \models c_i(x) \} \]
where $c_i$ is of the following form:

$$c_i = A \cup R.C \cup \forall R.C \cup \exists R.C \cup \neg n R.C \cup \neg n R.C$$

Here, $A$ is an atomic concept, $C$ is a class expression and $R$ is a role name in ontology $O$, and $m$ and $n$ are positive integers. $C$ is of the following form:

$$C = A \cup C_1 \cup C_2 \cup C_3 \cup \forall R.C_1 \cup \exists R.C_1$$

$$\leq n R.C_1 \cup \geq n R.C_1$$

where $C_1$ and $C_2$ are also class expressions.

Note that, in the above definition, the first-level expressions (the $c_i$s) are free from disjunctions. If an individual satisfies a disjunctive clause (a set of independent expressions combined using disjunctions), satisfiability of each of these independent expressions can be checked and if found $s$ included as conjunctions in the label-set. Clearly, the conjunction of all the elements in the label-set of an individual will be entailed by the ontology. That is, $O \models (\bigwedge_{i=1}^{n} c_i)(x)$

An example of the node-label-set of the individual $x = \text{tom}$ from ACAD ontology is: $L_O(x) = \{\text{Student, IITStudent, IIT_MS_Student, \exists hasAdvisor.TeachingStaff, \exists hasAdvisor.Professor, \\exists EnrolledIn.IITProgramme, \leq 1 hasAdvisor.TeachingStaff, \leq 1 hasAdvisor.Professor}\}$

**Edge-label-set** The label-set of a pair of individuals $(x, y)$ is the set that contains all the property relations (role names) from the first individual to the second individual. It is represented as $L_O(x,y)$.

**Definition 2** $L_O(x,y)$ is formally defined as (where $N_R$ is the set of all atomic roles in ontology $O$): $L_O(x,y) = \{R | R \in N_R \land O \models R(x,y)\}$.

From ACAD ontology, the edge-label-set of the pair $(\text{tom}, \text{bob})$ can be written as: $L_O(\text{tom}, \text{bob}) = \{\exists hasAdvisor\}.$

Although various approaches can be considered for generating label-sets, the practical method that we have adopted for generating the label-sets is explained in the next subsection.

### 4.2. Label-set generation technique

**Node-label-set generation.** The naive method to find the node-label-set of an individual is by doing satisfiability check for all combinations of roles, concepts and restrictions types; and include them if they are true. Since, this is not a practically adoptable method for large ontologies, we generate the label-set of an individual $x$ from an ontology $O$ as follows.

Firstly, we create the corresponding inferred ontology $O'$ (using a reasoner). From $O'$, we find all the concept names and (existential, universal and cardinality) restrictions satisfied by the individual as follows:

**Step 1:** All the concept names which are satisfied by $x$ are obtained by a simple SPARQL query. We can call it as the seed label-set. For example, the set of concept names, which we obtained from $O'$, corresponding to the individual $\text{tom}$ is $\{\text{Student, IITStudent, IIT_MS_Student}\}$.

**Step 2:** In order to get the restrictions satisfied by $x$, we access the class definitions and class subsumption axioms corresponding to the concepts which are obtained in the first step, and then consider the existential, universal and cardinality restrictions on the right hand side of those axioms to enrich the label-set.

The right hand side of the axioms in their conjunctive normal form (CNF) is used for enriching the label-set. That is, the R.H.S. will be of the form: $c_1 \cap c_2 \cap \cdots \cap c_k \cap \cdots \cap c_{k+n}$. Those clauses in the CNF which do not contain any disjunction, for examples as in $c_1, c_2$ etc. are directly included in the label-set. If a clause contains disjunction of expressions (denoted as D-Clause), such as $c_3 \lor c_4 \lor c_5 \lor \cdots \lor c_k$ above, then it is handled in parts, as shown in Algorithm[1].

Continuing with our example, enrichment of the label-set of $\text{tom}$ is done by obtaining existential, universal and cardinality restrictions associated with each of the concept names in the seed label-set. That is, the restrictions $\exists \text{EnrolledIn.IITProgramme},$
Algorithm 1 Handling disjunctions of expression

1: procedure LABEL-SET-GEN(x, D-Clause)  
2:     for each expression exp in D-Clause do  
3:         if exp is of the form ∃R.C then  
4:             if O |= ∃R.C(x) then  
5:                 \( L_O(x) \leftarrow L_O(x) \cup \{∃R.C\} \)  
6:             end if  
7:         else if exp is of the form ∀R.C then  
8:             if O |= ∀R.C(x) then  
9:                 \( L_O(x) \leftarrow L_O(x) \cup \{∀R.C\} \)  
10:            end if  
11:        else if exp is of the form ≤nR.C then  
12:            if O |= ≤nR.C(x) then  
13:               \( L_O(x) \leftarrow L_O(x) \cup \{≤nR.C\} \)  
14:            end if  
15:        else if exp is of the form ≥nR.C then  
16:            if O |= ≥nR.C(x) then  
17:               \( L_O(x) \leftarrow L_O(x) \cup \{≥nR.C\} \)  
18:            end if  
19:        end if  
20:     end for  
21: end procedure

For example, by looking at the node-label-set of tom, we will get the set of all restrictions (logical expressions) that are satisfied by the individual. Considering these restrictions together, we can frame a meaningful definition for tom as: “Tom is a student who is enrolled in an IIT Programme, has one professor as advisor, and all his advisors are teaching staffs.” Clearly, not all logical expressions (labels) in the label-set are necessary to generate such a description. That is, those labels that can induce redundancy in the description can be ignored or combined with other restrictions.

As noted earlier, some of the labels (mainly role restrictions) in the label-set if verbalized directly may generate confusing descriptions, and hence they should be reduced or combined with other restrictions to get a more refined restriction. For example, if left unreduced, the restrictions \( \text{VhasAdvisor.TeachingStaff} \) and \( \text{VhasAdvisor.T} \) may give rise to the description: “all advisors are some one and all advisors are teaching staffs”, which confuses a human reader.

Given a label-set, the naive method to remove redundant labels is by considering combinations of labels and trying to see whether they can be reduced or not. This is indeed a tedious process, since the total number of steps to be taken for complete reduction depends of the combination which we select at each step. To overcome this, we propose a rule-based process where labels of a specific restriction types are handled in a pre-defined order. A systematic method utilizing a set of rules which will always generate stricter (more specific) forms of a given set of restriction, is also proposed to attain complete refinement of the label-sets. Due to the aforementioned property of the rules, we call them as refinement-rules. Since we do this reduction or refinement of labels at the logical-level by considering their semantics, we call this rule-based refinement process as semantic-refinement of label-sets. The refined form of the label-set is called semantically-refined label-set.

The semantic-refinement is not only done to remove redundant labels in a label-set, but also to avoid ambiguous verbalization of interim logical expressions. For example, \( \text{VhasAdvisor.Professor} \) is a label which can appear in the label-set of an individual of IITStudent due to the axiom: IITStudent ⊑ VhasAdvisor.Professor. Linguistically this label (along with the axiom) can be interpreted in two ways. That is, either as All advisors of IIT students are teaching staffs or by considering logical equivalent of the statement, it can be interpreted as Either all advisors of IIT students are teaching staffs or (vacuously-true

≤ IITStudent hasAdvisor.TeachingStaff, ∀IITStudent hasAdvisor.TeachingStaff, and ∃IITStudent hasAdvisor.Professor associated with concept names are included in \( L_O(tom) \).

It should be noted that, using this approach, we are generating only those necessary restrictions which can entail the other satisfying combinations as per our label-set definition. For the same reason, we may need to rely on rule-based reasoning (explained later) to generate other restrictions which are of our interest.

Edge-label-set Generation The edge-label-set of a pair of individuals \((x, y)\) can be easily generated from \( O' \) using a simple SPARQL query.

5. Proposed Method for Generating Descriptions

Once we get the label-sets of all the individuals (node-label-sets) in a given ontology, we can generate descriptions of individuals and concepts using the following approaches.

5.1. Description of individuals

Node-label-sets of each individuals are considered for generating their descriptions. Label-sets of all the individuals from ACAD ontology is given in Table 5.
Case) they do not have an advisor. Clearly, including the latter description in the verbalization may confuse a reader. This is especially the case when it can be inferred from other axioms that vacuously-true case does not arise.

For identifying the cases where combinations of conditions involving qualifiers and/or number restrictions occur and to succinctly represent them, we introduce the following new constructors.

- Non-vacuous role restriction: \( R.C \)
  \[
  \exists R.C \Delta = \{ x \in \Delta | \exists y.(x, y) \in R^x \land y \in C^x \land \forall z.(x, z) \in R^z \implies z \in C^z \}
  \]

- Exactly-one role restriction: \( \exists_1 R.C \)
  \[
  \exists_1 R.C \Delta = \{ x \in \Delta | \exists y_1.(x, y_1) \in R^x \land y_1 \in C^x \land \exists y_2.(x, y_2) \in R^x \land y_2 \in C^x \implies y_1 = y_2 \}
  \]

- Exactly-\( n \) role restriction: \( \exists_n R.C \), general case of exactly-one role restriction.

In our rule-based refinement process, like any rule-based approach, the order at which the rules are applied is important, as the applicability of one rule may depend on another. We observed that there is a notion of strictness associated with role restrictions which can be effectively utilized for ordering the rules. The notion of strictness can be looked at as: if a role restriction \( R_1 \) is implied by another role restriction \( R_2 \) (i.e., \( R_2 \implies R_1 \)), then \( R_1 \) can be said as a stricter version of \( R_2 \). For instance, \( \exists R.U \) can be said as the stricter form of \( \forall R.U \) and \( \forall R.U \). Similarly, \( \exists_n R.U \) is a sticker form of \( \leq n R.U \) and \( \geq n R.U \). Since we intend to find sticker forms of role-restrictions, the obvious way is to apply rules corresponding to less stricter restriction types prior to those of stricter restriction types.

In the forthcoming sub-section, we introduce our rule-based refinement algorithm to accomplish complete reduction, where, we do all the possible reduction of less stricter restrictions prior to reducing stricter ones. Completeness of the refined form of label-set is guaranteed by the construction of the algorithm.

In what follows, we discuss how semantic-refinement of label-sets can be achieved.

### 5.1.1. Semantic-refinement of label-sets

We propose seven sets of rules for refining a label-set. Each of these rule sets contain carefully chosen rules which are repeatedly applied to the restrictions in the label-set, until no more reduction is possible. On moving from one rule set to another, those labels which have been reduced would be provisionally removed from the label-set. More details about the algorithm is given in the next sub-section.

The details of the first five sets of rules are given in Table 5. Each of the rule sets are given names that correspond to the type of restriction they handle. For example, the first rule set is called Concept Refinement rule, where atomic concepts in the label-set are looked at for refinement. More details about the refinement rules are given below.

- **Concept Refinement Rule.** Here, we consider all the concept name symbols that are present in the label-sets and, check whether their definitions (i.e., the set of restrictions which defines the concept) are included in the label-set. If the defining restrictions of a concept are present in the label-set, the concept name can be removed, since it is a redundant content.

- **Superclass Refinement Rule.** Consider the individuals given in Table 5, we can see that their label-sets contain all the concept names which they belong to. Some of the concepts in these label-sets are hierarchically related (in class - super-class relationship) in the ontology, resulting in redundant labels. For example, consider the label-set \( L_O(\text{tom}) \), it contains the

### Table 5

Node-Label-set of individuals in ACAD ontology (intentionally omitted \( \top \) class from the label-sets)

| Label-set | Description |
|-----------|-------------|
| \( L_O(\text{tom}) \) | \{Student, IITStudent, IIT_MS_Student, \exists hasAdvisor.TeachingStaff, \forall hasAdvisor.TeachingStaff, \exists hasAdvisor.Professor \} |
| \( L_O(\text{sam}) \) | \{Student, IITStudent, IITPhdStudent, \exists hasAdvisor.TeachingStaff, \forall hasAdvisor.TeachingStaff, \exists hasAdvisor.Professor \} |
| \( L_O(\text{bob}) \) | \{Professor, TeachingStaff\} |
| \( L_O(\text{alice}) \) | \{AssistantProf, TeachingStaff\} |
| \( L_O(\text{roy}) \) | \{Professor, TeachingStaff\} |
concepts IIT_MS_Student and IITStudent. Since it can be inferred from the concept IIT_MS_Student that tom is also a IITStudent, we can say that IITStudent is a redundant information (label) in the label-set. We remove such redundant labels by using most-specific concept notion. Also an individual may be present in 2 or more such subsumption concept chains. In each chain we need to use the most-specific concept.

(Not that, this refinement rule is applied only after the applications of the concept refinement rule – some specialized concepts may get removed while applying the rules in the first rule set, therefore, it does not always mean that a refined label-set contains only specialized concept names)

The presence of redundant concept names in a node-label-set is mainly because, we do a classification on the ontology prior to the label-set generation.

The upcoming rule sets are meant for reducing the various role restrictions allowed in a SHIQ ontology.

Existential Role Refinement rule. According to this rule, if a label-set contains two labels of the form: ∃R.U and ∃S.V, and if they satisfy the condition: U ⊑ V & R ⊑ S, then they can be refined to ∃R.U. In general, all these rules are defined such that given a refined form and the condition which have been used for refinement, the non-refined forms of the restriction(s) can be traced back. This means that, the refinement is done without affecting the semantics/meaning of the restrictions. Formally, the correctness of the rule can be proven as follows:

**Proof of Rule 3a.** Given an ontology O with R and S as its roles, and U and V are two of its concepts, and O |= U ⊑ V, R ⊑ S, then ∃R.U ∩ ∃S.V ⊑ ∃R.U. To prove this, let us consider an individual x ∈ ∃R.U ∩ ∃S.V, clear it implies x ∈ ∃R.U. Therefore ∃R.U ∩ ∃S.V ⊑ ∃R.U. Now, if x ∈ ∃R.U, it implies that there exist an arbitrary a, such that (x, a) ∈ R, a ∈ U. Since U ⊑ V, we can say that a ∈ V. It implies, x ∈ ∃S.V. Similarly, since R ⊑ S, (x, a) ∈ R ⇒ (x, a) ∈ S Therefore, ∃R.U ⊑ ∃R.U ∩ ∃S.V.

Universal Role Refinement rules. This rule set contains two rules which help in refining universal role restrictions. If a label-set contains two role restrictions of the form: ∀R.U and ∀S.V, universal role refinement rules can be applied if they satisfy the conditions of the rule. For example, if the label-set contains VhasAdvisor, Professor and VhasAdvisor. TeachingStaff, and if Professor ⊑ TeachingStaff, we can refine those restrictions to VhasAdvisor. Professor. The correctness of the two rules can be easily be proven as follows.

**Proof of Rule 4a.** Given an ontology O which entails U ⊑ V and S ⊑ R (where R and S are roles, and U and V are concepts), then ∀R.U ∩ ∀S.V ⊑ ∀R.U ∩ ∀S.V. Proving ∀R.U ∩ ∀S.V ⊑ ∀R.U ∩ ∀S.V is trivial since ∀S.V ⊑ ∀S.V (given, U ⊑ V). Now, let x ∈ ∃R.U ∩ ∃S.V, suppose (x, a) ∈ S where a is
an arbitrary individual. Since \( S \subseteq R, (x, a) \in R \). It implies \( a \in U \) (since \( x \in \forall R.U \)). Therefore, we get \( x \in \forall S.U \). Hence, \( \forall R.U \cap \forall S.V \subseteq \forall R.U \cap \forall S.U \).

**Proof of Rule 4b.** Given an ontology \( O \) which entails \( V \subseteq U \) (where \( R \) is a role and, \( U \) and \( V \) are concepts), then \( \forall R.U \cap \forall R.V \equiv \forall R.V \). Proving \( \forall R.U \cap \forall R.V \equiv \forall R.V \) is trivial, since the L.H.S. can be written as \( \forall R.(U \cap V) \), and it is equivalent to \( \forall R.V \), since \( V \subseteq U \).

Further in this section, we refrain from giving the proof of correctness of the rules in the succeeding rule sets. An appendix is provided at the end of the paper with all the required proofs.

**III & IV Combination rules.** In this rule set, we refine the existential and universal role restrictions which are present in the label-set.

The details of the next set of rule sets are given in Table 7.

**Qualified Number Restriction Refinement rules.** In this set there are four rules. Here we mainly try to refine qualified number restriction restrictions (of the form \( \leq n.R.U \) or \( \geq m.S.V \)) to stricter version of the same form or to a exactly-\( n \) restrictions.

**Exactly-\( n \) Role Restriction rules.** In this rule set, we reduce the exactly-\( n \) role restrictions which are generated using the preceding rule-sets. The rule set is named so because, this is the only rule set where we try to reduce exactly-\( n \) role restrictions.

### 5.1.2. Algorithm for semantic-refinement

As we mentioned before, semantic-refinement helps in refining restrictions, which are present in a label-set, to their stricter forms by combining them using a set of rules. The rules are applied sequentially from rule-set 1 to 7. While applying the rules, on moving from one rule-set to another, provisional removal of reduced restrictions is done to reduce computational complexity. In our algorithm, we will mark such restrictions as PRs (Provisionally Reduced ones), so that at a later stage we can remove them permanently from the label-set.

Algorithm-2 describes the steps that has to be followed for applying the rules. This algorithm works by taking pairs of restrictions from the label-set, and looking for the applicability of the rules. If a rule is applicable, the restrictions will be checked for the following set of conditions, to decide whether to resume the reduction or not. These conditions are followed mainly to ensure quick reduction.

**Condition-1.** No need to further reduce two provisionally reduced (PR) restrictions. (This is because, the rule-sets are designed in such a way that if a particular combination of restriction types is reduced by a rule in one rule-set, the same combination will not occur in the succeeding rule-sets)

**Condition-2.** If a rule combines two restrictions (\( R1 \) and \( R2 \)) and generates either \( R1 \) or \( R2 \), then that \( R1 \) or \( R2 \) should not be marked as a PR. (This is because, in the rules such as 4a, 5c, 6a etc., one of their antecedent term gets repeated in the consequent part of the rule to ensure reverse implication (i.e., for preserving semantics). On applying such rules, if the regenerated terms are marked as PR, they may get permanently removed during the course of the algorithm, which is not acceptable.)

**Condition-3.** If the restrictions of a particular form are not used in successive rule-sets, the PR restrictions of that form can be removed. (Either they can be removed after the applications of rules in all the rule-sets)
Algorithm 2 Semantic-refinement of label-sets

1: procedure SEMANTIC_REFINEMENT($L_0(x)$)
2:  Mark all $u \in L_0(x)$ as not PRs
3:  Apply Concept Refinement rule and remove appropriate concept names from $L_0(x)$
4:  $R \leftarrow$ Rule-sets 2-7 \& list of pre-defined rules
5:  for each rule-set $rs \in R$ do
6:    Set $M, REF \leftarrow \phi$
7:    for each $(u, v) \in L_0(x) \times L_0(x)$ AND $u \neq v$ do
8:      if (NOT(MARKED_AS_PR(u)) AND NOT(MARKED_AS.PR(v))) then
9:        for each $(r \in rs)$ do
10:          if $r$ is applicable on $(u, v)$ then
11:            $M \leftarrow$ APPLY_RULE$(r, u, v)$
12:            $L_0(x) \leftarrow L_0(x) \cup M$
13:            $REF \leftarrow$ REF $\cup$ $\{u, v\}$
14:            if $u \in M$ then
15:              $REF \leftarrow$ REF $\setminus \{u\}$
16:              end if
17:            if $v \in M$ then
18:              $REF \leftarrow$ REF $\setminus \{v\}$
19:              end if
20:          end if
21:        end for
22:      end if
23:    end for
24:    MARK_AS_PR(REF)
25:    $L_0(x) \leftarrow L_0(x) \cup REF$
26:    for each $u \in L_0(x)$ do
27:      if the restn. type of $u$ is not used in the successive rule-sets AND MARKED_AS.PR(u) then
28:        $L_0(x) \leftarrow L_0(x) \setminus \{u\}$
29:      end if
30:    end for
31:  end procedure

Fig. 1. Steps involved in the semantic-refinement of $L_0(x)$. Arrows represent the application of rules.

or they can be removed at specific points where it can be determined that they will not be used by any rules from then on. The latter is computationally efficient.

For illustration, let us consider the node-label-set of the individual $sam$. Fig shows the refinement steps and the rules in the rule sets which are used for the refinement. $L_0(x)$ is represented vertically. In the figure, the arrows represent the application of rules. Rule numbers are represented in italics. A refinement of two restrictions may sometimes result in more than one restrictions, to represent them, the arrows are followed by brace brackets (\{\ldots\}) to show the resultant restrictions.

Initially, the algorithm marks all the labels in the label-set as not PR. Then the algorithm looks for the applicability of the rule 1a (concept refinement rule). In the figure, $L_0(x)$ contains the labels IITStudent and IITPhdStudent whose definitions are present in the label-set. Therefore, the Rule 1a is applied on those labels and remove them from the label-set. In the algorithm, lines 5-31 take the rest of the rule-set one at a time, and look for possible application of rules on pairs of restrictions in the label-set. In our example label-set, since no rules in the rule-sets 2, 3, and 4 are applicable, we move to the rule set 5. Now, the algorithm applies the rule 5c on two of the restrictions as shown in the figure and refine them to the two restrictions given in the brackets. Application of a rule will be done only if the restrictions in the pair are not marked as PR (checked using the function MARKED_AS.PR()). The if condition in the line-8 of the algorithm will take care of this. After the application of a rule (using the function APPLY_RULE()), the details of the reduced restrictions will be stored in the set variable $REF$. Based on the condition-2, ap-
appropriate changes have to be done on the contents of REF (lines 14-20). Once all the possible rules in a particular rule set are applied, the reduced restrictions will be marked as PRs (lines 24). Once the algorithm considers all pairs of labels and checks them for the applicability of all the rules in the current rule-set, the condition-3 will be checked for possible permanent removal of the PRs. The entire process will be repeated for all the succeeding rule-sets.

Coming back to our example label-set, after the application of Rule 5c, one of the reduced restriction is marked as PR (represented using \( \checkmark \)), while the other restriction is not marked as PR due to the condition-2. On changing the rule-set, since no other rules in rule-set 5 are applicable, the one which is marked as PR can be permanently removed since the condition-3 is satisfied. In the forthcoming iterations of the for loop (line 5), rules in the rule-set 6 and 7 are applied in similar fashion. In the last iteration, we will get the most refined set of labels, along with a set of restrictions which are marked as PRs. The restrictions which are marked as PRs are removed to get the refined label-set.

Illustration of the usefulness of the approach. The usefulness of semantic refinement can be illustrated by looking at the sentences that can be generated from the node-label-set before and after refinement. Considering the original node-label-set, \( \text{sam} \) can be defined as “A student, an IIT student, an IIT PhD student, who is enrolled in an IIT programme, has more than two advisors who is a teaching staff, has less than one and at least one advisor who is a professor, and all advisors are teaching staff”.

By making use of the refined node-label-set, we can generate a smaller and easily-understandable definition: “A student who is enrolled in an IIT programme, has exactly one advisor who is a professor and has at least one more advisor who is a teaching staff but not a professor”. More examples and evaluation results to support the usefulness of this approach are presented in Section 7.

5.2. Description of Concepts

A concept can be defined in a similar fashion as that of an individual using label-sets. To generate the description of a concept, we introduce a new individual as its member. It is important that the new individual should be assigned as the member of only the concept whose definition has to be found. Now, label-set corresponding to this newly introduced individual is utilized to generated the concept’s definition. The rationale behind introducing a new individual is that, in order to find the definition of a concept (say Concept A), we only need restrictions which are associated with it and its super-classes. Considering an existing individual may result in a case where it may belong to concepts which are sub-classes of the concept A; this results in including the restrictions associated with the specific-classes also in the label-set, which is undesirable. Introducing a new individual will overcome this issue; in addition, the approach will even work smoothly for those concepts which do not have an individual.

Let us look at an illustration of generating definition of IITPhdStudent from ACAD ontology. At first, we introduce the individual \( \text{ips} \) as a member of IITPhdStudent. Now we will find the label-set of \( \text{ips} \).

We get \( \mathcal{L}_0(\text{ips}) \) as \{Student, IITStudent, IITPhdStudent, \( \exists \text{isEnrolledIn.IITProgramme}, \geq 2 \text{hasAdvisor.TeachingStaff}, \leq 1 \text{hasAdvisor.Professor, } \text{hasAdvisor.TeachingStaff}, \text{hasAdvisor.Professor}\} \)

In the next step, we remove the concept name, whose definition has to be found, from the obtained label-set. That is, \( \mathcal{L}_0(\text{ips})\setminus\{\text{IITPhdStudent}\}. \) This new label-set is semantically-refined and verbalized to get the redundant-free description of the concept.

Therefore, IITPhdStudent can be defined as:
\[
\{ \text{Student, hasAdvisor.Professor, } \exists \text{isEnrolledIn.IITProgramme, hasAdvisor.TeachingStaff, } \geq 1 \text{hasAdvisor.Professor.(TeachingStaff } \sqcap \neg \text{Professor})\}
\]

Even though this approach works well for those concepts whose (axiomatized) definitions contain only conjunctive clauses, it may generate incomplete descriptions when the definition contains a disjunctive clause. For example, if the definition of the concept IITStudent is of the form IITStudent \( \equiv \)}
Table 9
Refined node-label-sets of individuals in ACAD ontology

| Individual | Refined-label-set                                                                                                                                 |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| sam        | {Student, \exists isEnrolledIn.IITProgramme, \exists hasAdvisor.Professor, \neg hasAdvisor.TeachingStaff, \geq 1 hasAdvisor.(TeachingStaff \cap \neg Professor)} |
| tom        | {Student, \exists isEnrolledIn.IITProgramme, \exists hasAdvisor.Professor}                                                                 |
| bob        | {Professor}                                                                                                                                           |
| alice      | {AssistantProfessor}                                                                                                                                    |
| roy        | {Professor}                                                                                                                                           |

\exists isEnrolledIn.IITProgramme \cap (IITPhdStudent ∪ IIT_MS_Student), the label-set of a newly introduced individual of IITStudent (say, stud) should be {IITPhdStudent ∪ IIT_MS_Student, \exists isEnrolledIn.IITProgramme}. However, our current label-set generation method will not include disjunctive clauses as such in the label-set, instead it will look for the satisfiability of each of the expression in the disjunctive clause (that is, IITPhdStudent(stud) and IIT_MS_Student(stud)), and include them in the label-set, if they are true. But, for stud, they will not be true as we are not explicitly adding any other facts into the ontology other than IITStudent(stud). Therefore, we will get the label-set as (IITStudent, \exists isEnrolledIn.IITProgramme) which is an incomplete label-set of the concept. On doing the next steps – removing the concept name itself from the label-set, and doing a semantic-refinement over it – the incompleteness persists. To overcome issue, after semantic refinement step, we will enrich the refined label-set with the previously encountered disjunctive clause(s). That is, we get the new refined label-set of stud as {IITPhdStudent ∪ IIT_MS_Student, \exists isEnrolledIn.IITProgramme}.

6. Natural Language Descriptions from the Refined Label-sets

In this paper, prime focus is given for the generation of redundancy-free descriptions of ontology entities represented in the form of logical expressions. Appropriate NL sentence generation of these logical forms is yet to be fully explored. However, for the completeness of the paper, we present a simple method which we have adopted to generate NL descriptions of individuals and concepts from their refined label-sets.

NL description of an entity is defined as the set of NL fragments which describes the class names and role restrictions it satisfies. An example of a description of tom is:

```
tom: is a student, enrolled in at least one IIT programme, and has exactly one professor as advisor
```

We consider a template similar to the following regular expression (abbreviated as regex) for generating descriptions of individuals and concepts.

```
Individual/concept: ("is") (("a") ClassName (",” | "and")?)* (RoleRestriction (",” | "and")?)*
```

In the above regex, ClassName specifies the concept names in the label-set. We use the rdfs:label role values of the class names as the ClassName. If rdfs:label role is not available, the local names of the URIs are used as the ClassName. For RoleRestriction, the role restrictions in the label-set are utilized. The role restrictions are treated in parts. We first tokenize the role names in the constraints. Tokenizing includes word-segmentation and processing of camel-case, underscores, spaces, punctuations etc. Then, we identify and tag the verbs and nouns in the segmented phase — as R-verb, R-noun respectively — using the Natural Language Tool Kit.

In our studies, we have also tried out variants of these constraint-specific templates to further tune

---

4In the absence of a proper verb, the phrase “related to” is used in its place.

5Python NLTK: [http://www.nltk.org/](http://www.nltk.org/)
Table 10

Examples of the descriptions of individuals and concepts from PD, HP and GEO ontologies, generated using the proposed and as well as the traditional approaches

| Entity type | Proposed approach | Traditional approach | Ontology |
|-------------|-------------------|----------------------|----------|
| Indivl.     | _Bird cherry Oat Aphid:_ is a biotic-disorder, having at least one pest and all its factors are pests. | _Bird cherry Oat Aphid:_ is a disorder, bio-disorder, pest damage and insect damage. It is all the following: has as factor only pest-insect, has as factor only pest, has as factor only organism and has as factor something. | PD |
| Indivl.     | _Black Chaff:_ is a plant bacterioses, having at least one microorganism and all its factors are microorganism. | _Black Chaff:_ is a disorder, a biotic disorder and a plant bacterioses. It is all the following: has as factor bacterioses, has as factor only organism, has as factor at least 1 thing, has as factor only micro-organism. | PD |
| Concept     | _Mite Damage:_ is a biotic-disorder, having at least one mite pest and all its factors are mite pests. | _Mite Damage:_ is a disorder, a biotic-disorder and a pest damage. It is all the following: has as factor only organism, has as factor only pest, has as factor only mite pest, has as factor at least 1 thing. | PD |
| Indivl.     | _Hermione Granger:_ is a Hogwarts Student, a muggle, a gryffindor, having exactly one cat as pet. | _Hermione Granger:_ is a Hogwarts student, a student, a human, a muggle, a gryffindor. It is all the following: has a pet, has as pet a cat, has as pet only creature, has at least 1 creature, has at most 1 creature, as pet. | HP |
| Concept     | _Hogwarts Student:_ is a Student, is a Gryffindor or Hufflepuff or Ravenclaw or Slytherin, and having exactly one pet. | _Hogwarts Student:_ is a student, a human, is a Gryffindor or Hufflepuff or Ravenclaw or Slytherin. It is all the following: has a pet, has as pet only creatures, has at least 1 creature, has at most 1 creature. | HP |
| Indivl.     | _Hedwig:_ is an owl, is related to at least one Hogwarts student and only Hogwarts student, as pet. | _Hedwig:_ is an owl, a pet, a creature. It is all the following: is pet of only Hogwarts student, is pet of a Hogwarts student. | HP |
| Indivl.     | _Jersey:_ is a subnational entity, a government organization and is related to exactly one sovereign state as a member. | _Jersey:_ is a geopolitical dependency, an organization, a governmental organization, an Independent continuant, a subnational entity and is a member of exactly one sovereign state. | GEO |
| Indivl.     | _Florida:_ is a government organization, is related to at least one nation as a part, and is related to exactly one sovereign state as a member. | _Florida:_ is a major administrative subdivision, an organization, a governmental organization, an Independent continuant, a subnational entity. It is all the following: is a part of at least one nation, and is a member of exactly one sovereign state. | GEO |

7. Empirical Evaluation

We present two case studies to explore the applicability of the redundancy-free description of individuals and concepts in validating the domain knowledge. Rather than choosing an ontology under development, we study the cases of validating two previously built ontologies.

In the study, domain experts were presented with two representations of the same knowledge: one is by...
direct verbalization of the label-sets and the other is by verbalizing them after finding the corresponding refined label-sets. Direct verbalization of a label-set generates texts (or descriptions) which are similar to those texts which are produced by an existing ontology verbalizer — we call this method as traditional approach, and the other as the proposed approach. Examples for the description texts that are generated using the proposed approach and traditional approach, from the Plant Disease (PD) ontology $^6$ HarryPotter (HP) ontology $^7$ and Geographical Entity (GEO) ontologies are given in Table $^10$. One can clearly see that those descriptions which are generated using the proposed approach are compact, precise and easy-to-understand when compared to those which are generated using the traditional approach.

**Scope of the study.** We have done the empirical study mainly for two reasons. Firstly, for finding whether the process of semantic-refinement is helpful in generating useful texts for describing the ontology. For this purpose, the experts were asked to rate their degree of understanding of the knowledge in the scale: (1) poor; (2) medium; (3) Good.

Secondly, to measure the usefulness of the generated sentences (i.e., the descriptions of individuals and concepts) in validating the domain knowledge, domain experts were told to choose from the options: (1) Valid (2) Invalid (3) Don’t know (4) Cannot be determined. Significance of these options is that, if a participant is choosing the 4th option, it is likely that she finds it difficult to reach a conclusion on the validity of the sentence presented. In addition, feedbacks are collected from the experts to get suggestions on improving the system.

**Dataset used.** We used two ontologies for generating descriptions. The first ontology is Plant-Disease ontology (PD ontology) developed by International Center of Agricultural Research in the Dry Areas (ICARDA), and the second one is a synthetic ontology, Data structures and Algorithms (DSA) ontology, developed by ORG group $^9$ at IIT Madras $^{10}$. More details about these ontologies are available at our project website $^9$. The current version of PD ontology has 546 individuals, 105 concepts and 15 object properties. The DSA ontology has 333 individuals, 23 concepts, 33 object properties and 21 datatype properties.

**Experimental setup.** For each of the individuals and concepts in the two ontologies we have generated corresponding NL descriptions from their node label-sets as well as from their refined label-sets, using an implemented prototype of the system. Since manual evaluation of all the generated descriptions is difficult, a selected number of descriptions were utilized for the study. The set of descriptions of individuals for the study were selected by grouping the entire descriptions based on their label-sets and randomly choosing one individual’s description from each group. The set of descriptions of concepts were selected from those set of descriptions (generated from refined label-sets) which are highly different from their counterparts that are generated from their non-refined label-sets. From PD ontology, 31 descriptions of individuals and 10 descriptions of concepts have been considered for evaluation. Similarly, for DSA ontology, 14 descriptions of individuals and 17 descriptions of concepts were chosen for evaluation. Then, experts of the two domains were asked to review the verbalized descriptions. Majority ratings of the sentences were considered for finding the statistics.

**Expert selection.** Seven experts of plant disease areas and fourteen experts of data structures and algorithms were involved in the study. The seven experts of PD domain have either a masters degree or a doctorate degree in the plant disease or agriculture related areas. The fourteen experts of DSA domain have successfully completed the advanced data structures and algorithms course offered at IIT Madras.

### 7.1. Results and Discussions

Fig $^2$-$5$ show the statistics w.r.t. the ratings given by the domain experts. Based on these statistics, we have answered the following two questions.

#### 7.1.1. How does the semantic refinement help in improving the understandability of the verbalized knowledge?

The degree of understanding of each of these descriptions to the domain experts can be identified by looking at the ratings (i.e., poor, medium or good) which they had chosen during the empirical study. If there exists an ambiguity in the description (due to

---

1. http://wiki.plantontology.org/index.php/Plant_Disease_Ontology
2. https://sites.google.com/site/ontoworks/home
3. https://bitbucket.org/uamsdbmi/geographical-entity-ontology/src (last accessed: 27/11/2015)
4. https://sites.google.com/site/ontoworks/home
5. https://www.iitm.ac.in/
6. https://sites.google.com/site/ontoworks/projects
its verbatim fidelity to OWL statements), they are expected to choose poor or medium as the level of understanding. To confine the reasons for ambiguity to the fidelity to OWL constructs alone, possible (manual) grammatical error corrections had been done on the generated text — as we were not using any sophisticated NL generation techniques. Grammatical errors such as subject-verb agreement errors, verb tense errors, verb form errors, singular/plural noun ending errors and sentence structure errors had been corrected.

Fig. 2. Y-axis shows the count of descriptions of a particular rating which are generated using our proposed approach and the traditional approach from the PD ontology

Fig. 3. Y-axis shows the count of descriptions of a particular rating which are generated using our proposed approach and the traditional approach from the DSA ontology

Fig. 4 shows the overall responses which we received from the seven domain experts for the descriptions of PD ontology. We call it as the overall response because, ratings are calculated by looking at the majority responses; that is, only if a description is rated as ‘good’ by at least 4 participants, it will be considered as a good description; similar is the case with poor and medium ratings. The dotted-bars represent the count of the descriptions of a particular rating which are generated using the proposed approach and the stripped-bars denote the count of those which are generated using the traditional approach. Similarly, Fig 4 shows the statistics of the responses received for DSA ontology. For PD ontology, out of 41 descriptions which are generated using the proposed approach, 34 were rated as ‘good’, whereas for those which are generated using the traditional approach, only 6 out of 41 texts were rated as ‘good’. For DSA ontology, 24 out of 31 descriptions generated by proposed approach are ‘good’, only 11 descriptions that are generated using the traditional approach were rated as ‘good’. These results highlight the significance of the semantic-refinement process in domain knowledge understanding.

Fig. 4. Statistics (based on the majority responses) to determine the usefulness of the generated descriptions in validating the PD ontology

Fig. 5. Statistics (based on the majority responses) to determine the usefulness of the generated descriptions in validating the DSA ontology

7.1.2. How does the semantic refinement helpful in knowledge validation?

Fig. 4 and 5 show the statistics to determine the usefulness of the generated descriptions in validating the two domain ontologies, where, as before, the dotted-bars represent the ratings of the descriptions that are generated from the proposed approach and the stripped-bars denote rating of the descriptions gener-
ated by the traditional approach. Usefulness of the generated descriptions in validating an ontology are obtained by looking at the number of descriptions which are marked as ‘Cannot be determined’. The three options: Valid, Invalid and Don’t know, imply that the text is useful in getting into a conclusion, whereas the option ‘Cannot be determined’ indicates that there is some problem in the representation. From Fig 4 and Fig 5 in case of the proposed approach, only 7 out of 41 descriptions from PD ontology and 4 out of 31 descriptions from DSA ontology were not useful in determining the quality of the ontology, whereas in case of the traditional approach, approximately 50 percentage of the descriptions were not helpful. This clearly indicates that, verbalization after semantic-refinement is more effective in applications such as ontology validation.

7.1.3. Discussion

The participants of our empirical study agree with the fact that, by reducing the redundancies in a description, the amount of time required for validating an individual description is reduced to a great extent.

Validation of an ontology also involves verifying the truthfulness of the property relationships in it, which is not addressed in this paper. This issue can be addressed in future by making use of the edge-label-sets (label-sets for pairs of individuals – see Section 4.1) and mapping them to the respective constraint(s) in the node-label-set of the first individual. For e.g., $L_\mathcal{O}(a) = \{C_1,C_2, \exists\text{hasFriend},C_3\}$, and $L_\mathcal{O}(a,b) = \{\text{hasFriend}\}$, then $\text{hasFriend}$ in $L_\mathcal{O}(a,b)$ can be mapped to $\exists\text{hasFriend},C_3$ in $L_\mathcal{O}(a)$. The description of $a$ can be generated as ‘$a$: is a $C_1$ and $C_2$, and has some $C_3$, like $b$, as Friend.’ Further investigation has to be done in this direction.

According to the domain experts, a persisting problem with any validation phase (especially when it involves descriptions ontology entities and experts validating the verbalized knowledge) is that, when the ontology becomes very large and complex, validation phase becomes a bottleneck for the entire development cycle. One way to overcome this issue is by considering only a relevant subset of individuals and concepts and their descriptions for validity check, so that, a rough estimate of the erroneous formalisms in the ontology can be identified quickly. Another direction of future work is a study on the order at which the descriptions are to be presented to an expert so that an early detection of invalid knowledge can be made possible.

8. Conclusion

A novel method for verbalizing the definitions (called natural language descriptions) of ontology entities is presented in the paper. The descriptions are not merely verbatim translations of logical axioms of the ontology. Instead, they are generated from the set of logical restrictions satisfied by individuals and concepts of the ontology on which semantic simplification had been carried out. We propose a rule-based reduction approach for this purpose. We find that the proposed method indeed gives redundancy-free descriptions of individuals and concepts.

Our empirical studies based on two ontologies have shown that the redundancy-free description of the domain knowledge is helpful in understanding the formalized knowledge more effectively and also useful in validating them.

Acknowledgements

This project is funded by Ministry of Human Resource Development, Gov. of India. We express our fullest gratitude to the participants of our evaluation process: Dr. S.Gnanasambadan (Director of Plant Protection, Quarantine & Storage), Ministry of Agriculture, Gov. of India; Mr. J. Delince and Mr. J. M. Samraj, Department of Social Sciences AC & RI, Killikulam, Tamil Nadu, India; Ms. Deepthi.S (Deputy Manager), Vegetable and Fruit Promotion Council Keralam (VFPCK), Kerala, India; Dr. K.Sreekumar (Professor) and students, College of Agriculture, Vellayani, Trivandrum, Kerala, India. We also thank all the undergraduate and post-graduate students of Indian Institute of Technology, Madras, who have participated in the empirical study.

References

[1] Ion Androutsopoulos, Gerasimos Lampouras, and Dimitrios Galanis. Generating natural language descriptions from OWL ontologies: the naturalowl system. CoRR, abs/1405.6164, 2014.
[2] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. The description logic handbook: theory, implementation, and applications. Cambridge University Press, New York, NY, USA, 2003.
[3] Anne Cregan, Rolf Schwitter, and Thomas Meyer. Sydney owl syntax - towards a controlled natural language syntax for owl 1.1. In Christine Golbreich, Aditya Kalyanpur, and Bijan Par-
Appendix A

Proofs for the rules in the rule-sets 5 to 7

Here we use proof-by-contradiction as the proof method. Given a rule of the form \( P \equiv Q \), we prove \( P \equiv Q \) by negating it and proving \( (P \cap \neg Q) \cup (Q \cap \neg P) \) as false.

Consider that all the following rules are defined on an ontology \( O \) with \( R \) and \( S \) as its roles, and \( U \) and \( V \) are two of its concepts.

Rule 5a: Given the ontology \( O \), \( \exists R.U \cap \forall R.U \equiv \exists R.U \). The proof is trivial, and can be easily derived from the definition of \( \exists R.U \).

Rule 5b: If \( O \vDash U \subseteq V \), \( S \subseteq R \), then for \( \forall R.U \cap \exists S.V \equiv \exists R.U \cap \exists S.U \).

Assume that \( \forall R.U \cap \exists S.V \cap (\exists R.U \cap \forall R.U) \cap (\exists S.U \cap \forall S.U) \) is true. We can write it as: \( \forall R.U \cap \exists S.V \cap (\exists R.U \cap \forall R.U) \cap (\exists S.U \cap \forall S.U) \Rightarrow \forall R.U \cap \exists S.U \cap \exists S.V \cap (\forall R.U \cap \exists S.U \cup \forall S.U \cup \exists S.U \cup \exists S.U) \) (since \( S \subseteq R \), \( \forall R.U \Rightarrow \forall S.U \) \( (\forall R.U \cap \exists S.U \cup \forall S.U \cup \exists S.U \cup \exists S.U) \)).

Now, assume that \( \neg (\forall R.U \cap \exists S.U \cap \forall S.U) \) is true. \( \equiv (\forall R.U \cap \exists S.U \cap \forall S.U) \). 

Assume that \( \forall R.U \cap \exists S.V \cap (\exists R.U \cap \forall S.U) \cap (\forall S.U \cap \exists S.U \cap \forall S.U) \cap (\exists S.U \cap \forall S.U) \). 

Now, assume that \( (\exists S.U \cap \forall S.U) \cap (\forall S.U \cap \exists S.U \cap \forall S.U) \). 

Rule 6a: If \( O \vDash U \subseteq V \), then for \( n \geq m \), \( \exists n.R.U \cap \forall n.S.V \equiv \exists n.R.U \).

Assume that \( \exists n.R.U \cap \forall n.S.V \cap \neg (\exists n.R.U) \) is true. We can write it as: \( \exists n.R.U \cap \forall n.S.V \cap \neg (\exists n.R.U) \Rightarrow \exists n.R.U \cap \forall n.S.V \cap \neg (\exists n.R.U) \).

Now, assume that \( \neg (\exists n.R.U \cap \forall n.S.V) \cap \neg (\exists n.R.U \cap \forall n.S.V) \cap \neg (\exists n.R.U \cap \forall n.S.V) \).

Rule 6b: If \( O \vDash V \subseteq U \), \( S \subseteq R \), then for \( n \geq 1 \), \( \exists R.U \cap \forall n.S.V \equiv \forall n.S.V \).

Assume that \( n \geq 1 \), \( \exists R.U \cap \forall n.S.V \cap \neg (\exists n.R.U) \) is true. We can write it as: \( \exists R.U \cap \forall n.S.V \cap \neg (\exists n.R.U) \Rightarrow \exists R.U \cap \forall n.S.V \cap \neg (\exists n.R.U) \).

Now, assume that \( \neg (\exists n.S.V \cap \forall n.R.U \cap \neg (\exists n.R.U \cap \forall n.S.V) \cap \neg (\exists n.R.U \cap \forall n.S.V) \cap \neg (\exists n.R.U \cap \forall n.S.V) \).
Rule 6c: If $O \models U \subseteq V$, then for $n = 1$, $3R.U \cap \neg nR.V \equiv 3nR.U \cap 3nR.V$.
Assume that, $3R.U \cap \neg nR.V \cap (\neg 3nR.U \cap 3nR.V)$ is true. We can write it as: $3R.U \cap \neg nR.V \cap (\neg (3R.U \cap \neg nR.V \cap (\neg 3nR.U \cap 3nR.V))) \equiv (3R.U \cap \neg nR.V \cap (\neg 3nR.U \cap 3nR.V) \cap (\neg (3R.U \cap \neg nR.V \cap (\neg 3nR.U \cap 3nR.V))))$.

Contradiction. The contradiction in the first conjunctive expression is because: $\geq 0.S.V \implies 3S.V \iff 3R.U$ which contradicts with $\forall R.\neg U$.

Rule 7b: If $O \models U \subseteq V, R \subseteq S$, then $3R.U \equiv 3nR.U \cap 3S.V \cap \neg R.U$.
Assume that, $3R.U \equiv 3nR.U \cap 3S.V \cap \neg (3nR.U \cap 3nS.V)$ is true. We can write it as: $3nR.U \equiv 3nS.V \cap \neg (3nR.U \cap 3nS.V)$.

Contradiction. The contradiction in the first conjunctive expression is because: $\geq 1.U \iff \geq 1.S.V \iff \geq 1.R.V$ which contradicts with $\exists nR.U$.

Now assume that $\exists nR.U \cap 3S.V \cap \neg (3nR.U \cap 3nS.V)$, now, $(\exists nR.U \cap 3S.V \cap \neg (3nR.U \cap 3nS.V)) \equiv (\exists nR.U \cap 3S.V \cap \neg (3nR.U \cap 3nS.V))$.

Contradiction. The contradiction in the first conjunctive expression is because: $\geq 0.S.V \iff \geq 0.R.V \iff \geq 0.V$ which contradicts with $\exists nR.U$.

Now assume that $\exists nR.U \cap 3nS.V \cap 3R.U \equiv 3nR.U \cap 3nS.V \cap \neg (3nR.U \cap 3nS.V)$ is true.

$\equiv 3nR.U \equiv 3nS.V \equiv (\neg 3nR.U \cap 3nS.V) \equiv (\exists nR.U \cap 3nS.V \cap \neg (3nR.U \cap 3nS.V)) \equiv (\exists nR.U \cap 3nS.V \cap \neg (3nR.U \cap 3nS.V))$.

Contradiction. The contradiction in the first conjunctive expression is because: $\geq 0.S.V \iff \geq 0.R.V \iff \geq 0.V$ which contradicts with $\exists nR.U$.

Now assume that $\exists nR.U \cap 3nS.V \cap 3R.U \equiv 3nR.U \cap 3nR.U \equiv \neg (3nR.U \cap 3nS.V)$, now, $(\exists nR.U \cap 3nS.V \cap 3R.U \equiv 3nR.U \cap 3nR.U \equiv \neg (3nR.U \cap 3nS.V))$.

Contradiction. The contradiction in the first conjunctive expression is because: $\geq 0.S.V \iff \geq 0.R.V \iff \geq 0.V$ which contradicts with $\exists nR.U$.

Now assume that $\exists nR.U \cap 3nS.V \cap 3R.U \equiv 3nR.U \cap 3nR.U \equiv \neg (3nR.U \cap 3nS.V)$, now, $(\exists nR.U \cap 3nS.V \cap 3R.U \equiv 3nR.U \cap 3nR.U \equiv \neg (3nR.U \cap 3nS.V))$.

Contradiction. The contradiction in the first conjunctive expression is because: $\geq 0.S.V \iff \geq 0.R.V \iff \geq 0.V$ which contradicts with $\exists nR.U$.

Now assume that $\exists nR.U \cap 3nS.V \cap 3R.U \equiv 3nR.U \cap 3nR.U \equiv \neg (3nR.U \cap 3nS.V)$, now, $(\exists nR.U \cap 3nS.V \cap 3R.U \equiv 3nR.U \cap 3nR.U \equiv \neg (3nR.U \cap 3nS.V))$.

Contradiction. The contradiction in the first conjunctive expression is because: $\geq 0.S.V \iff \geq 0.R.V \iff \geq 0.V$ which contradicts with $\exists nR.U$.
\[
\equiv \exists_{\equiv n} R. U \cap \geq (m-n) R. (V \cup \neg U) \cap (\neg (\exists_{\equiv n} R. U) \sqcup \leq m-1 R. V)
\]

In the second conjunctive clause, contradiction can be found as follows: an \( x \in \geq (m-n) R. (V \cup \neg U) \) implies \( x \) has more than \( m-n \) \( R \) relations to \( \neg U \cap V \), since \( x \in \exists_{\equiv n} R. U \), we can say that \( x \) has more than \( m-n + n \) \( R \) relations to \( V \), which can be written as \( x \in \geq m R. V \). Clearly, this contradicts with \( \leq (m-1) R. V \).