Semantic Authoring of Ontologies by Exploration and Elimination of Possible Worlds

Sébastien Ferré

Semantic Authoring of Ontologies by Exploration and Elimination of Possible Worlds

Sébastien Ferré*

IRISA, Université de Rennes 1
Campus de Beaulieu, 35042 Rennes, France
Email: ferre@irisa.fr

Abstract. We propose a novel approach to ontology authoring that is centered on semantics rather than on syntax. Instead of writing axioms formalizing a domain, the expert is invited to explore the possible worlds of her ontology, and to eliminate those that do not conform to her knowledge. Each elimination generates an axiom that is automatically derived from the explored situation. We have implemented the approach in prototype PEW (Possible World Explorer), and conducted a user study comparing it to Protégé. The results show that more axioms are produced with PEW, without making more errors. More importantly, the produced ontologies are more complete, and hence more deductively powerful, because more negative constraints are expressed.

1 Introduction

Ontology authoring is generally an essential step in the application of knowledge engineering, and Semantic Web technologies. Existing methodologies generally distinguish two phases: (a) *conceptualization*, and (b) *formalization* in a formal language, typically the Web Ontology Language (OWL) [7]. We are here concerned with the formalization phase, which presents a number of difficulties, in particular for beginners but not only. Some difficulties are related to the manipulation of a formal language. Tools like Protégé [10] have precisely been introduced to facilitate such manipulation. Other difficulties are related to the discrepancies that can arise between the original intention of the ontology author, and what the formal ontology really express [12,3]. For example, "only eats vegetables" does not imply "eats some vegetables"; or to know that "X is a woman" does not allow to infer that "X is not a man" unless it has been explicitely stated that "men and women form disjoint classes". Indeed, negative constraints, like class disjointness or inequalities between individuals, are often overlooked because they seem so obvious. Their omission is difficult to detect because they do not manifest themselves by erroneous inferences, but by missing inferences. In a previous paper [6], we have shown errors and important omissions in the Pizza ontology[1], albeit it is used as a model and pedagogical support for years.

* This research is supported by ANR project IDFRAud (ANR-14-CE28-0012-02).

[1] http://protege.stanford.edu/ontologies/pizza/pizza.owl

For example, classes `Food` and `Country` are not disjoint, and it appears that a vegetarian pizza can actually contain meat and/or fish as ingredient.

We introduce a new approach to ontology authoring that is centered on semantics rather than on syntax. Rather than seeing an ontology as a set of axioms, we propose to see an ontology through its set of models, i.e. as the set of interpretations that satisfy the ontology. We informally call those models "possible worlds". In the same spirit, rather than seeing ontology authoring as the successive addition of axioms, we propose to see it as the successive elimination of "possible worlds". Each elimination of a subset of "possible worlds" generates an axiom so that we still obtain a set of axioms in the end. However, the generated axioms are only the result of the authoring process, not the means. The main advantage of this approach is to enable the ontology author to work at the level of instances – possible worlds – like for ontology population (particular knowledge), while actually defining the terminological level of the ontology (general knowledge). From a previous paper [6], we reuse *possible world exploration*, and the contribution of this paper is to support the *creation* of an ontology from scratch rather than the mere *completion* of an existing ontology. Another contribution is a user study comparing our prototype PEW to Protégé.

Section 2 presents related work on ontology authoring. Section 3 recalls the basics of description logics, and Section 4 recalls previous results about possible world exploration, and prototype PEW. Section 5 presents the extension of PEW for ontology authoring, and Section 6 sketches an example scenario for the formalization of hand anatomy. Section 7 details the methodology and results of our user study. Section 8 concludes with a few perspectives.

2 Related Work

Ontology editors such as Protégé [10] tend to favor the expression of positive constraints, i.e. axioms supporting the inference of positive facts: e.g., class hierarchy, domain and range of properties. Their users have a mostly syntactic view of their ontology, and are hardly exposed to their semantics. By semantics, we here mean which situations the ontology makes possible or not. Semantic feedback can be obtained by calling a reasoner to check the consistency of the ontology, or the satisfiability of a class expression. However, those calls are tedious and left to the users. OntoTrack [9] offers a graphical view of the ontology, and returns immediate semantic feedback when the ontology is modified. However, it only covers a fragment of OWL Lite, and the view is limited to class hierarchies. The use of *competency questions* has also been proposed [13] to specify what the ontology is expected to answer, and then to automatically test the ontology during authoring. To some extent, the exploratory approach of our work enables to generate and validate at the same time such competency questions through interaction. Another reasoner-assisted ontology authoring approach [8] adapts test-driven development from softwares to ontologies by defining for each type of axiom a test to be run before and after the insertion of each axiom.

Table 1. Syntax and semantics of some DL class and property constructors, followed by TBox and ABox axioms. Thereby C, D denote class expressions, R, S property expressions, a, b individual names, and r a property name.

Name	Syntax	Semantics	
top	$\top$	$\Delta^{\mathcal{I}}$	
bottom	$\bot$	$\emptyset$	
negation	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$	
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$	
disjunction	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$	
nominal	$\{a\}$	$\{a^{\mathcal{I}}\}$	
exist. restriction	$\exists R.C$	$\{x \in \Delta^{\mathcal{I}} \mid \text{for some } y \in \Delta^{\mathcal{I}}, (x,y) \in R^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\}$	
univ. restriction	$\forall R.C$	$\{x \in \Delta^{\mathcal{I}} \mid \text{for all } y \in \Delta^{\mathcal{I}}, (x,y) \in R^{\mathcal{I}} \text{ implies } y \in C^{\mathcal{I}}\}$	
inverse property	r^-	$\{(y,x) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (x,y) \in r^{\mathcal{I}}\}$	
subclass	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$	TBox axioms
subproperty	$R \sqsubseteq S$	$R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$	
instance	$C(a)$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$	ABox axioms
relation	$R(a,b)$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$	
same	$a \doteq b$	$a^{\mathcal{I}} = b^{\mathcal{I}}$	
different	$a \neq b$	$a^{\mathcal{I}} \neq b^{\mathcal{I}}$	

A number of controlled natural languages, such as CLOnE [4] or Rabbit [5] have been proposed to produce OWL axioms from sentences in natural language, and to verbalize OWL ontologies in natural language. They address the issue about the syntax of formal languages, not the issue about semantic feedback. Their contribution is therefore orthogonal to ours, and could complement it.

There also exists a number of (semi-)automated techniques to produce OWL axioms. Some tools detect common errors, and complete ontologies in a systematic way [11,3]. However, those approaches are not constructive but corrective. Moreover, they are often limited to disjointness axioms, the simplest form of negative constraints. Formal Concept Analysis [1,14] has been used in interactive ontology authoring. Experts are presented with a sequence of statements, and for each of them, they have to either confirm the statement, or produce a counter-example. It guarantees complete formalization for some DL fragments but it is rather expensive in terms of user interaction, and tends to patronize the expert.

3 Preliminaries

We here recall basic definitions of Description Logics (DL), which are the basis of OWL ontologies [7]. We briefly recap here the syntax and semantics of the sublanguage of OWL DL that is necessary for this work. We assume finite and disjoint sets N_I, N_C, and N_R, respectively called *individual names*, *class names* and *property names*. Table 1 shows how complex classes, complex properties, and axioms can be formed from these atomic entities. An *ontology* $\mathcal{O}$ is a set of axioms, which are often partioned in two subsets: a TBox containing general

axioms about classes and roles, and an ABox containing particular axioms about individuals. The semantics of description logics is defined via interpretations $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ composed of a non-empty set $\Delta^{\mathcal{I}}$ called the *domain of* $\mathcal{I}$ and a function $\cdot^{\mathcal{I}}$ mapping individuals to elements of $\Delta^{\mathcal{I}}$, classes to subsets of $\Delta^{\mathcal{I}}$ and properties to subsets of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ (i.e., binary relations). This mapping is extended to complex classes and properties, and finally used to evaluate axioms (see Semantics in Table 1). We say $\mathcal{I}$ satisfies an ontology $\mathcal{O}$ (or $\mathcal{I}$ is a model of $\mathcal{O}$, written: $\mathcal{I} \models \mathcal{O}$) if it satisfies all its axioms. We say that an ontology $\mathcal{O}$ *entails* an axiom α (written $\mathcal{O} \models \alpha$) if all models of $\mathcal{O}$ are models of α. Finally, an ontology is *consistent* if it has a model and a class C is called *satisfiable* w.r.t. an ontology $\mathcal{O}$ if there is a model $\mathcal{I}$ of $\mathcal{O}$ with $C^{\mathcal{I}} \neq \emptyset$.

4 Possible World Exploration

We here recall from a previous paper [6] an approach for a safe and complete exploration of the possible worlds of an OWL DL ontology. No assumption is made on the ontology. It may contain instances or not. It may be limited to taxonomies or contain complex DL axioms. The exploration is based on navigation, where navigation places are situations made possible by the ontology (formally, satisfiable class expressions): e.g., "pizzas without topping" in the Pizza ontology. Navigation links enable to move from one place to another, i.e. from one situation to another (formally, transformations of class expressions). A key aspect is that those navigation links are automatically computed for each situation so as to never lead users to impossible situations. If we see the models $\mathcal{I}$ of an ontology $\mathcal{O}$ as "possible worlds", then each navigation place offers a view on a subset of possible worlds. If a navigation place is a situation described by the class expression C, then the subset of possible worlds is made of models $\mathcal{I}$ of the ontology that make the class expression satisfiable ($C^{\mathcal{I}} \neq \emptyset$). Therefore, navigation across situations supports in effect the exploration of possible worlds.

A prototype, PEW (Possible World Explorer), is available as open source[2]. It relies on HermiT [15] for all reasoning tasks. For the sake of concision, we here use the DL notation (see Section 3) for axioms and class expressions. In practice, PEW gives users the choice between the DL notation and the Manchester notation. In the following, examples are based on the Pizza ontology.

4.1 Views over Possible Worlds

At every navigation step, a view over the possible worlds selected by the current situation is presented to the ontology author. In order to formally define those views, we first define two sublanguages of class expressions. *Simple class expressions* serve as elementary situation descriptions, and *cognitively intuitive class expressions* combine them to describe complex situations. We called the latter "cognitively intuitive" because they restrict negation to simple class expressions, which make situations easier to grasp for humans. Indeed, "humans would

[2] http://www.irisa.fr/LIS/softwares/pew

normally have no problems with handling the class of non-smokers or childless persons, while classes such as non-(persons having a big dog and a small cat) occur unnatural, contrived and are harder to cognitively deal with"[6].

Definition 1. *Given sets N_C, N_R, N_I of class names, property names, and individual names, the set $\mathcal{S}$ of* simple class expressions *is the set of class expressions of one of the forms (with $A \in N_C$, $r \in N_R$, $a \in N_I$): A, $\{a\}$, $\exists r.\top$, $\exists r.A$, $\exists r.\{a\}$, $\exists r^-.\top$, $\exists r^-.A$, $\exists r^-.\{a\}$, and their negations $\neg A$, $\neg\{a\}$, $\neg\exists r.\top$, etc.*

Definition 2. *The set $\mathcal{CI}$ of* cognitively intuitive class expressions *is inductively defined as follows:*
1. every simple class expression is in $\mathcal{CI}$,
2. for $C_1, C_2 \in \mathcal{CI}$, $C_1 \sqcap C_2$ and $C_1 \sqcup C_2$ are in $\mathcal{CI}$,
3. for any property r and $C \in \mathcal{CI}$, $\exists r.C$ and $\exists r^-.C$ are in $\mathcal{CI}$.
The set $\mathcal{CI}[X]$ of pointed $\mathcal{CI}$ class expressions *denotes $\mathcal{CI}$ class expressions with symbol X occurring exactly once in the place of a subexpression.*

The situations that can be described by $\mathcal{CI}$ class expressions involve existing objects, their identity, their membership to atomic classes, and their interrelationships. The syntax and semantics of description logics entail that only tree-shape relationships can be expressed. Additionally, disjunction enables to express alternatives for some parts of the situation.

A pointed class expression $C(X) \in \mathcal{CI}[X]$ is used to put a *focus* on a subexpression of a class expression. It is a class expression with a hole in place of a subexpression, materialized by the meta-variable X: e.g., $C(X) = \mathit{Pizza} \sqcap \exists \mathit{ingredient}.X$ represents a pizza with an unspecified ingredient. Given a class expression D, the expression $C(D)$ denotes the expression $C(X)$ where D has been substituted for X, filling the hole. For example, given $D = \mathit{Meat} \sqcup \mathit{Fish}$, we have $C(D) = \mathit{Pizza} \sqcap \exists \mathit{ingredient}.(\mathit{Meat} \sqcup \mathit{Fish})$.

Definition 3. *Given an ontology $\mathcal{O}$, a* possible world view *$C(X)/D$ is specified by the combination of a pointed class expression $C(X)$ (the* context*), and a class expression D (the* focus*), such that $C(D)$ is a satisfiable class in $\mathcal{O}$. The focus is said* necessary *if $C(\neg D)$ is not satisfiable. A view is also composed of* instances *and* adjuncts*, derived from context and focus:*

- *$\mathit{Inst}(C(X)/D) = \{a \in N_I \mid \mathcal{O} \models C(D)(a)\}$ is the set of instances of $C(D)$;*
- *$\mathit{Adj}(C(X)/D) = \{E \in \mathcal{S} \mid C(D \sqcap E)$ is satisfiable in $\mathcal{O}\}$ is the set of* possible adjuncts *at focus. A positive simple expression E is called a* necessary adjunct *if $E \in \mathit{Adj}$ and $\neg E \notin \mathit{Adj}$*[3].

In a view, the expression $C(D)$ represents the situation, and its decomposition into context and focus enables the representation of the focus. That focus is essential for possible world exploration as it enables to look at the different

[3] When disjunctions are used, the definitions of satisfiable class and adjuncts are slighty more complex to ensure that all alternatives remain satisfiable. See [6] for details.

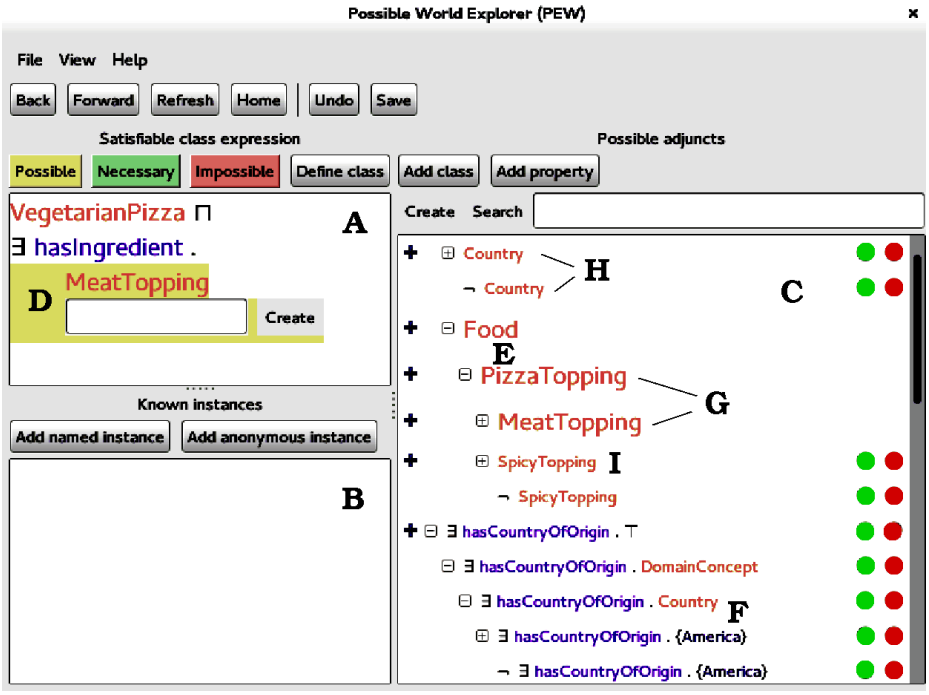

Fig. 1. Screenshot of PEW: looking at vegetarian pizzas with meat.

objects of the situation. Only situations described by satisfiable classes, aka. "possible worlds", are considered. When the expression under focus cannot be negated in the context, the focus is said necessary. In the Pizza ontology, the view $Pizza \sqcap X/\exists hasIngredient.\top$ has a necessary focus because every pizza has necessarily an ingredient ($\mathcal{O} \models Pizza \sqsubseteq \exists hasIngredient.\top$). Therefore, PEW allows to check any class inclusion $C \sqsubseteq D$ by exploration when $C, D \in \mathcal{CI}$.

The instances of a view are the individuals that are a member of the class $C(D)$ in all models of the ontology. The adjuncts of a view are the simple class expressions E that are satisfied by the object at the focus of the situation in at least one model of the ontology. This is checked by evaluating the satisfiability of the situation after inserting E at the focus: $C(D \sqcap E)$. There are three cases for each positive simple class expression E. If E and $\neg E$ are possible adjuncts, we call E "ambivalent". If only E is possible, we call it "necessary". If only $\neg E$ is possible, we call it "impossible". The case where both E and $\neg E$ are not possible adjuncts is excluded because it would imply that $C(D)$ is not satisfiable.

Figure 1 shows a screenshot of PEW. The user interface reflects the definition of views with the class expression at the top left (**A**), the instances at the bottom left (**B**), and the possible adjuncts at the right (**C**). The focus subexpression is highlighted by a background color (**D**) that is normally yel-

low, and becomes green when the focus is necessary. The described situation is here a "vegetarian pizza that has meat as an ingredient", with focus on the "meat". In the example, there are no known instance of the situation. The possible adjuncts are organized in a tree to reflect the class hierarchy, the property hierarchy, and the membership of individuals to classes. For example, we see that *PizzaTopping* is a subclass of *Food* (**E**), and that *America* is a member of *Country* (**F**). Necessary adjuncts are shown in a larger font (**G**), and impossible adjuncts are not displayed to avoid cluttering the interface. Ambivalent adjuncts appear as pairs $E, \neg E$, e.g. *Country*, $\neg$*Country* (**H**). The unexpected facts that the screenshot tells us about the possible worlds of the Pizza ontology are that: (1) a vegetarian pizza can have meat as ingredient (**A**, **D**), and (2) a meat topping can be a country or not (**H**). Fact (1) is possible because vegetarian pizzas are defined to exclude meat as topping, but not as ingredient. Fact (2) is possible because a disjointness axiom between *Food* and *Country* is missing. The screenshot also shows reasonable facts: a meat topping can be spicy (**I**), and can have a country of origin, for example America (**F**).

4.2 Moving in the Space of Possible Worlds

The principle of possible world exploration is to move from situation to situation, in order to look at different corners of the space of possible worlds. Moving to another situation is done by applying transformations to the class expression (context and focus) that represents the current situation.

Definition 4. *Let $V = C(X)/D$ be a view. The available transformations are:*

Inserting an adjunct. *Choosing a possible adjunct $E \in Adj(V)$, set context as $C(D \sqcap X)$, and set focus as E.*
Inserting a disjunction. *Set context as $C(D \sqcup X)$, and set focus as $\top$.*
Deleting focus subexpression. *Keep context as $C(X)$, and set focus as $\top$.*
Moving the focus. *Considering the class expression $C(D)$, and choosing a subexpression D', set focus as D', and set context $C'(X)$ as the class expression $C(D)$ where D' has been replaced by X.*

The most important transformations are "inserting an adjunct" to extend the situation description, and "moving the focus" to choose where to extend that description. An important result is the *safeness* and *completeness* of possible world exploration, i.e. all and only possible and cognitively intuitive situation can be reached through navigation (see proofs in [6]).

Theorem 1. *Starting from the initial view $X/\top$ defined by context $C(X) = X$, and focus $D = \top$ ($C(D) = \top$), every reachable view has a satisfiable class expression (*safeness*), and every view defined by a satisfiable class expression in $\mathcal{CI}$ is reachable in a finite number of navigation steps (*completeness*).*

For example, the situation of Figure 1 is reached by the successive insertion of two possible adjuncts: *VegetarianPizza*, $\exists hasIngredient.MeatTopping$. In PEW, inserting an adjunct is done by double-clicking it, and moving the focus is done by clicking on the desired subexpression. Inserting a disjunction or deleting the focus are done through the contextual menu.

5 Possible World Elimination for Ontology Authoring

Compared to our previous work, PEW now offers a wide range of *commands* to update an ontology. Previously, the only command was to make the current situation impossible, generating the axiom $C(D) \sqsubseteq \bot$. Although many axioms can be rewritten in that form, there were important limitations. First, the signature was fixed, limiting its application to the completion of existing ontologies, and forbidding the creation of new ontologies. Second, the restriction to atomic negations limited the expressivity of positive constraints such as class inclusion $C \sqsubseteq D$. Indeed, the latter axiom is equivalent to $C \sqcap \neg D \sqsubseteq \bot$, and therefore the right class expression D was limited to simple class expressions. Third, a number of simple axioms, while expressible, required a contrived formulation, and many navigation steps. For example, to state that class A is a subclass of B, it was necessary to select adjunct A, then select adjunct $\neg B$, and finally declare the resulting situation impossible. The following update commands are designed to minimize the use of explicit negation from the user point of view.

Update commands. Let $C(X)/D$ be a possible world view. The current version of PEW supports the following commands to update the ontology.

New class. Extending the ontology signature with a new class name A.
New property. Extending the ontology signature with a new property name r.
Add instance. Extending the ontology signature with a new individual a, and making it an instance of the class expression with axiom $(C(D))(a)$.
Add subclass of class adjunct B. Creating a new class name A, and making it a subclass of B with axiom $A \sqsubseteq B$.
Add subproperty from relational adjunct $\exists s.\top$. Creating a new property r, and making it a subproperty of s with property axiom $r \sqsubseteq s$.
Add inverse property from relation adjunct $\exists s.\top$. Creating a new property r, and making it the inverse of s with axiom $r \equiv s^{-}$.
Add property constraint from relational adjunct $\exists s.\top$. Constrain property s to be any of functional, inverse functional, reflexive, irreflexive, symmetric, asymmetric, and/or transitive.
Impossible situation. Add axiom $C(D) \sqsubseteq \bot$.
Necessary focus. Add axiom $C(\neg D) \sqsubseteq \bot$. It has no effect if the focus is already necessary.
Impossible adjunct E. Add axiom $C(D \sqcap E) \sqsubseteq \bot$. This is equivalent to first inserting adjunct E at focus, and then triggering command "Impossible situation". It is forbidden on necessary adjuncts.
Necessary adjunct E. Add axiom $C(D \sqcap \neg E) \sqsubseteq \bot$. This is equivalent to first inserting adjunct E at focus, moving focus on E, and then triggering command "Necessary situation". It is useless on necessary adjuncts.
Disjoint adjuncts $E_1, \ldots E_n$. Add axiom $E_i \sqcap E_j \sqsubseteq \bot$ for each pair $\{E_i, E_j\}$. This command is equivalent to $\frac{n(n-1)}{2}$ "impossible situation" commands.

The main controls that trigger update commands are visible in **Figure 1**. Above the class expression (**A**), there are buttons to make the current situation

Table 2. Translation of main DL axioms into PEW views and commands. Axioms are restricted to cognitively intuitive class expressions ($C, D \in \mathcal{CI}$).

DL axiom		PEW view	PEW update command
TBox	$C \sqsubseteq D$	$(C \sqcap X)/D$	necessary focus
	$R \sqsubseteq S$	$X/\top$	add subproperty R from relation adjunct $\exists S.\top$
ABox	$C(a)$	X/C	add instance a
	$r(a, b)$	$X/\{a\}$	necessary adjunct $\exists r.\{b\}$
	$a \doteq b$	$X/\{a\}$	necessary adjunct $\{b\}$
	$a \neq b$	$X/\{a\}$	impossible adjunct $\{b\}$

"impossible" or to make the current focus "necessary". Button "possible" has the effect to declare new classes, properties and individuals that may have been inserted directly in the class expression through the entry field in (**D**). Button "define class" allows to declare a new class name A, and to make it equivalent to the class expression ($A \equiv C(D)$). Above the instances (**B**), the first button allows to "add an instance to the class expression". The second button does the same with an anonymous individual. Above the tree of adjuncts (**C**), the two buttons allow to extend the signatures with "new classes and properties". Each positive adjunct has a blue cross on its left to "add subclasses" (when the adjunct is a class name), and "add subproperties" (when the adjunct is an existential restriction). Each possible adjunct has on its right a green dot to make it "a necessary adjunct", and a red dot to make it "an impossible adjunct". Other commands are available through the contextual menu of the tree of adjuncts. A general principle of the user interface is to provide as much immediate feedback as possible. For instance, when the focus is made necessary, its color switches from yellow to green; when an ambivalent adjunct is made necessary, the negative adjunct disappears, and the positive adjunct switches to a larger font.

To assess the expressivity of PEW, Table 2 explains for each kind of axiom how to express it by specifying which view to reach by navigation, and which update command to trigger. Note that both TBox and ABox axioms are covered. The only restriction is that built class expressions must be cognitively intuitive, i.e. must have only negation on simple class expressions. Note that, compared to the previous version, command "necessary focus" enables class inclusions with complex class expressions on both sides. PEW is therefore *complete* w.r.t. the chosen sublanguage of OWL. The features of OWL2 that are not covered are literals and related features, cardinality restrictions, local reflexivity (*self*), property chains, and keys.

Similarly to possible adjuncts that are computed so as to avoid the construction of unsatisfiable class expressions, update commands are designed to avoid the production of inconsistencies in the ontology. If a command would produce an inconsistency, it is blocked and an error message is shown to the user. Commands "impossible adjunct" and "necessary adjunct" are only available on ambivalent adjuncts because they are either useless or inconsistent on necessary and impossible adjuncts.

6 Example Scenario: Ontology of Hand Anatomy

We here sketch a possible strategy to formalize the basics of the anatomy of a normal hand. (We strongly recommend the reader to watch the 8min screencast at https://youtu.be/u4X0hq6et0Q.) First, new classes are created for the different types of elements of the hand: *Hand*, *Palm*, *Finger*, *Phalanx*, and *Nail*. Those 5 classes are made disjoint. Then, 5 subclasses are added to *Finger* for each finger (thumb, index, middle, ring, and little), and 3 subclasses to *Phalanx* for each phalanx (proximal, middle, and distal). Those two sets of subclasses are also made disjoint. The next step is to create property *hasDirectPart*, and its inverse *isDirectPartOf*. From there, the ontology signature is complete, and completing the ontology is a matter of exploring the possible worlds, and triggering "impossible" and "necessary" commands. For example, looking at hands with view $X/Hand$, the possible adjuncts show that at this stage a hand can contain any element, and be part of any element. Here, most possible adjuncts are either made necessary (e.g., $\exists HasDirectPart.IndexFinger$) or impossible (e.g., $\exists isDirectPartOf.Hand$). Some possible adjuncts may remain ambivalent, e.g. $\exists isDirectPartOf.\top$, to let open the possibility to make hands part of larger elements (e.g., arms). The following steps are to visit in turn each element type, similarly to hands, to apply the relevant constraints: e.g., distal phalanges have nails, proximal phalanges have no nail, thumbs have no middle phalanx. The domain of property r can be constrained by reaching view $X/\exists r.\top$, and its range by reaching view $(\exists r.X)/\top$, which is only one focus move from the former view.

7 User Study: Comparison with Protégé

We conducted a user study to compare PEW and Protégé in the task of designing an ontology from scratch. The ontology to be designed is the same as in the previous section, on hand anatomy. The advantages of this topic are that the related knowledge is well known to everybody, and that it is rich with many positive and negative constraints.

7.1 Methodology

The subjects of the user study are 30 postgraduate students in bioinformatics (Master of Bio-Informatics and Genomics at Université Rennes 1). Before the user study, subjects had been exposed to a short course on Semantic Web technologies (RDF, RDFS, OWL); but they had been exposed neither to Protégé nor to PEW. The user study was organized as a practical about OWL ontologies for the Semantic Web course. Students were simply told that their work would be used for a research experiment. Subjects were cast in two groups, one working with PEW, and the other one working with Protégé. Like in all practicals, students worked either alone or in a pair. In the following, we refer to the singleton or pair of students working together as a *team*. In total, 9 teams worked with PEW, and 8 teams worked with Protégé.

The user study lasted about 2h, starting with 20min of presentation and tutorial about the system of the group, ending with 10min to fill a SUS questionnaire [2], and having a maximum of 1h30min to design the requested ontology on hand anatomy. A document was distributed to each team with instructions, requirements, and a row of questions to guide the design of the ontology in a progressive and modular way. At the end of the sessions, we collected the OWL file of each team, and the anonymous SUS questionnaire filled by each student. The distributed document, the collected data, and analysis spreadsheets are available at http://www.irisa.fr/LIS/ferre/pub/ekaw2016.zip.

7.2 Results

Quantity and correction. For each team, we counted the produced OWL declarations and axioms, and among them the number of erroneous axioms. We did not count uninformative or imprecise axioms as errors, only axioms that are inconsistent with the anatomy of a normal hand. The table below reports for each system the minimum, average, and maximum value of those measures across teams. It also reports the resulting precision of produced axioms: the individual precision is the average of precisions team-wise, while the collective precision is computed on the collection of all axioms produced with a system. The result is that more axioms were produced with PEW: 74% increase on average, and nearly three-fold at maximum. The next paragraph shows that the increase is mostly due to one kind of axioms. Despite that increase, we do not observe more errors produced with PEW. This entails a higher precision for PEW, with a significant difference for collective precision. The fact that collective precision is higher than individual precision for PEW, and the inverse for Protégé says that the PEW teams that produced the more axioms made less errors than the average, and that, on the contrary, the Protégé teams that produced the more axioms made more errors.

system	nb. declarations/axioms	nb. errors	individual precision	collective precision
Protégé	43 (68) 88	0 (5) 17	77% (93%) 100%	92%
PEW	58 (118) 241	0 (4) 18	69% (95%) 100%	97%

Axiom types. We analyzed the type of produced axioms, and counted for each team the number of axioms of each type. In addition to declarations and property axioms (e.g., inverse property), we distinguish between positive axioms in the form $C \sqsubseteq D$, and negative axioms in the form $C \sqsubseteq \neg D$ or equivalently $C \sqcap D \sqsubseteq \bot$. Then, we consider all combinations between atomic classes (A, B) and simple qualified existential restrictions ($\exists r.A$, $\exists r.B$), except combinations with two restrictions because they are rare and mostly errors. We decomposed the produced axioms to make them fit the previous types, when possible. For example, axiom $A \equiv B$ splits into $A \sqsubseteq B$ and $B \sqsubseteq A$; and an axiom declaring A as the domain of R translates to $\exists R.\top \sqsubseteq A$. For cardinality restrictions produced with Protégé, we counted minimum cardinality as an existential restriction, and maximum cardinality as a negated existential restriction. Figure 2 compares the average number of axioms produced by teams for each system, and for each type

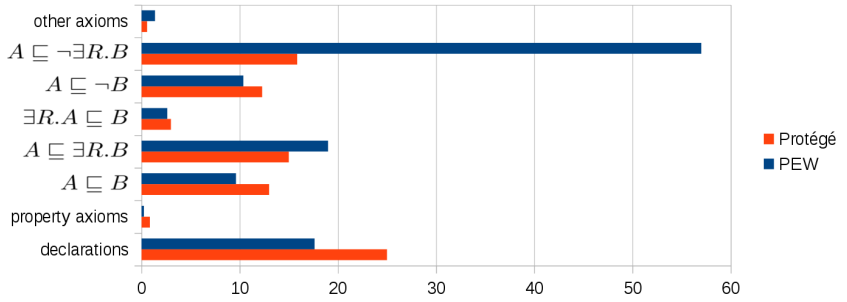

Fig. 2. Comparison of the average number of declarations and axioms per type.

Table 3. Sample of constraints (12 positive, 8 negative) that must be satisfied by a normal hand. They are organized in 5 types of DL axioms.

id	DL axiom	informal description
	$A \sqsubseteq B$	*subclass relationship*
P1	$IndexFinger \sqsubseteq Finger$	every index finger is a finger
P2	$ProximalPhalanx \sqsubseteq Phalanx$	every proximal phalanx is a phalanx
	$A \sqsubseteq \exists R.B$	*relation existence*
P3	$Hand \sqsubseteq \exists hasPart.IndexFinger$	every hand has an index finger
P4	$IndexFinger \sqsubseteq \exists isPartOf.Hand$	every index finger is part of a hand
P5	$Finger \sqsubseteq \exists hasPart.ProximalPhalanx$	every finger has a proximal phalanx
P6	$IndexFinger \sqsubseteq \exists hasPart.MiddlePhalanx$	every index finger has a middle phalanx
P7	$ProximalPhalanx \sqsubseteq \exists isPartOf.Finger$	every proximal phalanx is part of a finger
P8	$DistalPhalanx \sqsubseteq \exists hasPart.Nail$	every distal phalanx has a nail
P9	$Nail \sqsubseteq \exists isPartOf.DistalPhalanx$	every nail is part of a distal phalanx
	$\exists R.A \sqsubseteq B$	*qualified property domain/range*
P10	$\exists hasPart.Palm \sqsubseteq Hand$	only hands have a palm as a direct part
P11	$\exists isPartOf.Finger \sqsubseteq Phalanx$	only phalanges are parts of fingers
P12	$\exists hasPart.Nail \sqsubseteq DistalPhalanx$	only distal phalanges have a nail
	$A \sqsubseteq \neg B$	*class disjointness*
N1	$Hand \sqsubseteq \neg Finger$	no hand is a finger
N2	$Thumb \sqsubseteq \neg IndexFinger$	no thumb is an index finger
N3	$ProximalPhalanx \sqsubseteq \neg DistalPhalanx$	no proximal phalanx is a distal phalanx
	$A \sqsubseteq \neg\exists R.B$	*relation non-existence*
N4	$Hand \sqsubseteq \neg\exists hasPart.Hand$	no hand is made of a hand
N5	$Hand \sqsubseteq \neg\exists isPartOf.Finger$	no hand is part of a finger
N6	$Thumb \sqsubseteq \neg\exists hasPart.MiddlePhalanx$	no thumb has a middle phalanx
N7	$ProximalPhalanx \sqsubseteq \neg\exists isPartOf.Phalanx$	no proximal phalanx is part of a phalanx
N8	$ProximalPhalanx \sqsubseteq \neg\exists hasPart.Nail$	no proximal phalanx has a nail

of axiom. The striking one difference is about complex negative axioms in the form $A \sqsubseteq \neg\exists R.B$, which were produced 3.5 times more often with PEW. A typical example is $Thumb \sqsubseteq \neg\exists hasPart.MiddlePhalanx$ stating that the thumb has no middle phalanx. An interesting example of complex axiom that was produced with PEW is $\exists orientation^{-}.\exists isPartOf.\exists orientation.\{RIGHT\} \sqsubseteq \{RIGHT\}$, stating that every part of a right element is a right element.

Recall estimate. Another question we wanted to answer is about the completeness of the produced ontologies. Indeed, producing more axioms does not imply a more complete formalization of the domain. Intuitively, measuring com-

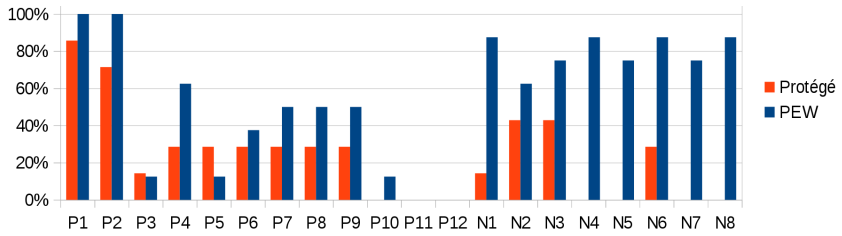

Fig. 3. Comparison of recall on 20 constraints: 12 pos. (P1-P12), 8 neg. (N1-N8).

pleteness amounts to counting the proportion of constraints that are entailed by the ontology. To make it practical, we have listed 20 constraints represented as DL axioms (see Table 3), and evaluated the recall of each ontology $\mathcal{O}$ over them as an estimate for completeness, i.e. counting axioms α s.t. $\mathcal{O} \models \alpha$. We have chosen the 20 constraints to cover all above types of positive and negative axioms, and to have a representative coverage of the basics of hand anatomy (types of hand elements, and part-of relationships between elements). The table below gives the minimum, average, and maximal values of recall for both systems, and for three sets of constraints: all of them, only positive ones, only negative ones. Globally, ontologies produced with PEW were more than twice as complete on average, and even 75% complete in the best case. In fact, the least complete PEW ontology is still more complete than the average Protégé ontology. That difference is even stronger for negative constraints, and remains to a lesser degree for positive constraints. For negative constraints, PEW average recall reaches 80%, 5 times higher than Protégé average recall (16%). On negative constraints, the least complete PEW ontology (50%) is still neatly more complete than the most complete Protégé ontology (38%).

system	all constraints	pos. constraints	neg. constraints
Protégé	0% (24%) 45%	0% (29%) 50%	0% (16%) 38%
PEW	35% (56%) 75%	17% (41%) 58%	50% (80%) 100%

Figure 3 compares the average recall for each constraint. All negative constraints have a recall above 63% with PEW, and below 43% with Protégé.

SUS questionnaire. The subjective perception of students about the systems was collected through the classic SUS questionnaire [2]. We got 13 answers for each system. The results show only small differences between the two systems. The score for PEW is 52 on average, ranging from 25 to 85, and is slightly better for Protégé, 58 on average, ranging from 18 to 80. Students tend to find PEW more complicated (*"I think that I would need the support of a technical person to be able to use this system"* 3.8 vs 3.1) but with less inconsistencies (*"I thought there was too much inconsistency in this system"* 2.2 vs 2.5) than Protégé, although the differences are small. Looking at extreme votes (1 and 5 on a 1-5 scale), it appears that PEW triggers more contrasted opinions with some students finding it easy to use, and others finding it very complicated. An interesting comment

by a PEW user says that the difficulty was about the syntax of class expressions (in Manchester syntax), rather than about the tool itself.

7.3 Interpretation and Discussion

PEW is more productive. We think that this is because in Protégé, there is a high step between the axioms that are easy to express in the interface, such as class hierarchy, class disjointness, domains and ranges, and other axioms that require to actually write class expressions. In PEW, there is also a step between simple class expressions that are readily available as possible adjuncts, and more complex class expressions that require several navigation steps. However, simple class expressions offer more expressivity than the easy axioms of Protégé, and navigating to complex class expressions is arguably simpler than writing them.

PEW achieves a better precision. Errors in OWL axioms often come from cognitively unintuitive constructors, in particular universal restriction and general negation. For example, a Protégé team produced axiom $\mathit{Hand} \sqsubseteq \forall \mathit{hasPart}.\mathit{RingFinger}$, using $\forall$ instead of $\exists$. The restriction in PEW to cognitively intuitive class expressions implies that users are only asked to evaluate factual (i.e., existential and positive) situations as impossible or necessary.

PEW achieves a better recall, especially when there are many negative constraints. PEW presents a symmetry between positive and negative constraints because they are produced in the same way, just using command "necessary" for the former, and "impossible" for the later. On the contrary, Protégé has a strong bias towards positive constraints, apart from class disjointness. However, negative constraints are essential to the deductive potential of expressive ontologies.

PEW usability is encouraging. PEW is a prototype that is much less mature than Protégé but it got a SUS score not far behind Protégé. Its bad reception by a few proves that the design of the user interface must be improved. The good reception by a few others shows that there is ample room for such improvement. The main issue is the readability of class expressions. Another issue is the fact that commands in contextual menus (e.g., inverse property) are often overlooked compared to commands as buttons.

8 Conclusion

We have presented a semantic approach to ontology authoring based on possible world exploration and elimination. It has been implemented as prototype PEW, and a user study has demonstrated promising results in terms of quantity, precision, and recall of the produced axioms, and in terms of usability. The most notable result is the increase in recall, from 24% with Protégé to 56% with PEW, where 100% would mean a complete OWL formalization of the domain knowledge for the selected OWL fragment. Future work will investigate long-run guidance for the systematic exploration of possible worlds in order to

further improve recall; and the verbalization in natural language of class expressions to improve the readability of explored situations so as to further improve precision. We also plan to re-implement PEW as a Protégé plugin in order to combine their strengths, and favor its adoption.

Acknowledgement. We wish to thank Olivier Dameron for his precious support in the user study, as well as the students of master BIG for their kind participation.

References

1. Baader, F., Ganter, B., Sertkaya, B., Sattler, U.: Completing description logic knowledge bases using formal concept analysis. In: Int. Joint Conf. Artificial Intelligence. pp. 230–235 (2007)
2. Brooke, J.: SUS: A quick and dirty usability scale. In: Jordan, P., Thomas, B., Weerdmeester, B., McClelland, A. (eds.) Usability evaluation in industry, pp. 189–194. London: Taylor and Francis (1996)
3. Corman, J.: Explorer les théorèmes d'une TBox. In: Journées francophones d'Ingénierie des Connaissances (2013)
4. Davis, B., Iqbal, A.A., Funk, A., Tablan, V., Bontcheva, K., Cunningham, H., Handschuh, S.: Roundtrip ontology authoring. In: International Semantic Web Conference. pp. 50–65. Springer (2008)
5. Denaux, R., Dimitrova, V., Cohn, A.G., Dolbear, C., Hart, G.: Rabbit to OWL: ontology authoring with a CNL-based tool. In: Int. Work. Controlled Natural Language. pp. 246–264. Springer (2009)
6. Ferré, S., Rudolph, S.: Advocatus diaboli - exploratory enrichment of ontologies with negative constraints. In: ten Teije et al., A. (ed.) Int. Conf. Knowledge Engineering and Knowledge Management. pp. 42–56. LNAI 7603, Springer (2012)
7. Hitzler, P., Krötzsch, M., Rudolph, S.: Foundations of Semantic Web Technologies. Chapman & Hall/CRC (2009)
8. Keet, C., Lawrynowicz, A.: Test-driven development of ontologies. In: et al., H.S. (ed.) Extended Semantic Web Conf. (ESWC). LNCS 9678, Springer (2016)
9. Liebig, T., Noppens, O.: OntoTrack: A semantic approach for ontology authoring. Web Semantics 3(2), 116–131 (2005)
10. Noy, N., Sintek, M., Decker, S., Crubezy, M., Fergerson, R., Musen, M.: Creating semantic web contents with Protege-2000. Intelligent Systems, IEEE 16(2), 60–71 (2001)
11. Poveda-Villalón, M., Suárez-Figueroa, M., Gómez-Pérez, A.: Validating ontologies with OOPS! In: Knowledge Engineering and Knowledge Management (EKAW), pp. 267–281. Springer (2012)
12. Rector, A., Drummond, N., Horridge, M., Rogers, J., Knublauch, H., Stevens, R., Wang, H., Wroe, C.: OWL pizzas: Practical experience of teaching OWL-DL: Common errors & common patterns. In: Engineering Knowledge in the Age of the Semantic Web, pp. 63–81. Springer (2004)
13. Ren, Y., Parvizi, A., Mellish, C., Pan, J.Z., Van Deemter, K., Stevens, R.: Towards competency question-driven ontology authoring. In: European Semantic Web Conference. pp. 752–767. Springer (2014)
14. Rudolph, S.: Acquiring generalized domain-range restrictions. In: Int. Conf. Formal Concept Analysis. pp. 32–45. LNCS 4933, Springer (2008)
15. Shearer, R., Motik, B., Horrocks, I.: Hermit: A highly-efficient OWL reasoner. In: OWLED. vol. 432 (2008)