Advocatus Diaboli - Exploratory Enrichment of Ontologies with Negative Constraints

Sébastien Ferré, Sebastian Rudolph

Advocatus Diaboli – Exploratory Enrichment of Ontologies with Negative Constraints

Sébastien Ferré[1] and Sebastian Rudolph[2]

[1] IRISA, Université Rennes 1, France,
Sebastien.Ferre@irisa.fr
[2] KIT, Karlsruhe, Germany
Sebastian.Rudolph@kit.edu

Abstract. With the persistent deployment of ontological specifications in practice and the increasing size of the deployed ontologies, methodologies for ontology engineering are becoming more and more important. In particular, the specification of negative constraints is often neglected by the human expert, whereas they are crucial for increasing an ontology's deductive potential. We propose a novel, arguably cognitively advantageous methodology for identifying and adding missing negative constraints to an existing ontology. To this end, a domain expert navigates through the space of satisfiable class expressions with the aim of finding absurd ones, which then can be forbidden by adding a respective constraint to the ontology. We give the formal foundations of our approach, provide an implementation, called Possible World Explorer (PEW) and illustrate its usability by describing prototypical navigation paths using the example of the well-known pizza ontology.

1 Introduction

Ontologies – logical descriptions of a domain of interest – are at the core of Semantic Technologies. Expressive ontology languages like OWL allow for very precise specifications of semantic interdependencies between the notions describing a domain of interest. While it has been argued that "a little semantics goes a long way" and lightweight formalisms provide for better scalability properties, it is also widely accepted that expressive formalisms are superior in terms of modeling power and the capability of deriving implicit knowledge, thereby allowing for a more intelligent way of handling information.

From the viewpoint of formal semantics, the axioms of an OWL ontology can be seen as conditions or constraints which a possible world has to satisfy for being in line with what the ontology modeler has specified to be "true" in the considered domain. Thereby, one can distinguish between *positive constraints*, which specify what must be necessarily true, and *negative constraints* declaring what is impossible.[3] It has often been noted that positive constraints – such as

[3] Note that, on this general level, the distinction is conceptual rather than technical: for instance, the positive constraint that all catholic priests must be unmarried can likewise be read as the negative constraint that the existence of a married catholic priest is impossible.

class memberships, class and role hierarchies, or domain and range restrictions – are more salient and graspable to human beings and are preferably specified by modelers, whereas typical negative constraints – like class or role disjointness – are often neglected. However, negative constraints are crucial for exploiting the full deductive potential of expressive ontological modeling. In particular, they are essential for causing inconsistencies, being a helpful feature for many ontology management tasks, e.g. detecting modeling errors in the course of ontology creation and refinement [6], repairing mappings between two ontologies [9] or revising ontologies interactively [12].

In order to overcome this problem, many automated techniques have been designed to extract negative constraints from ontologies themselves or other sources, such as texts [17, 18, 10, 4]. The majority of these techniques rely on heuristics and machine learning methods whence their results are not entirely reliable and usually need to be inspected manually. Moreover, the mentioned approaches are restricted to disjointness, the simplest form of negative constraints. On another note, in the course of interactive ontology completion strategies based on Formal Concept Analysis [14, 1, 15], negative constraints are naturally acquired next to positive ones. As a downside, these techniques are rather expensive in terms of user interaction and tend to patronize the expert by forcing her to just answer a prescribed row of questions.

We propose to approach the problem from a different angle by providing more freedom to the domain expert and representing the task of specifying negative constraints in a cognitively apt (and, hopefully, interesting or even playful) way. This is achieved by letting the expert navigate the possibilities left open by the currently specified ontology, using a faceted browsing approach, and discover absurd configurations. In a sense, the modeler explores the "Platonic universe" where everything ontologically possible also exists. This way, configurations which are practically impossible can be identified, turned into negative constraints, and added to the ontology.

The paper is structured as follows. In Section 2, we lay out the basic idea of our methodology. Section 3 provides syntax and semantics of the description logic underlying OWL in a condensed way, while Section 4 introduces further formal notions needed for our approach. Section 5 describes our navigation approach on a technical level and establishes properties which ensure its adequacy. In Section 6, we introduce the Possible World Explorer (PEW), a tool which implements the proposed methodology and in Section 7 we illustrate its usefulness and usability by describing exemplary runs of it. Section 8 concludes and describes avenues for future work. An extended version of this paper including full formal proofs is available as technical report [3].

2 The Advocatus Diaboli Methodology

Here we give a non-technical overview of our envisioned methodology for the specification of negative constraints by exploring possible worlds.

As a starting point, we assume that an OWL ontology has been created by stipulating the used vocabulary and possibly arranging classes and properties in taxonomies. It is not essential that the ontology contains individuals, our method works equally well for non-populated ontologies. Also, the ontology may or may not already contain axioms beyond taxonomic relationships.

According to the model-theoretic semantics, an ontology can be seen as a set of constraints characterizing possible worlds (the models of the ontology). Adding axioms to an ontology results in strengthening these constraints and thereby "ruling out" models.

Our methodology can be seen as an exploration of the possible worlds admitted by an ontology. Thereby, a domain expert starts to describe an individual of one of these possible worlds by specifying its class memberships and relationships to other individuals. This specification process is supported by an interface in the spirit of faceted browsing, which naturally constrains the specification process in a way that no descriptions can be constructed which would contradict the ontology. In other words, the navigation-like stepwise refinement of the description of a possible individual ensures that an individual matching this description indeed exists in at least one of the models of the ontology. In this sense, the proposed methodology can indeed be seen as an "exploration of possible worlds".

The actual task of the domain expert is now to construct descriptions which are possible according to the ontology but absurd given the experts domain knowledge. That is, the domain expert is supposed to assume the role of the "devils advocate" by actively trying to construct situations which are impossible according to his/her knowledge of the domain, thereby showing that the given ontology is underconstrained. Once such a problematic description has been constructed, it can be converted into an axiom which exactly prohibits this situation. By adding this axiom to the ontology, the just constructed absurd description is made impossible and every model featuring such a situation is excluded from the possible worlds.

From a cognitive viewpoint, the particular twist of this methodology is that it facilitates the specification of negative constraints by staying on a positive, scenario-like level, by exploring what is (logically) possible, pushing the boundaries of what is conceivable, and trying to cross them by constructing "nonsensical" descriptions. Arguably, this is much easier and more intuitive than the task of directly coming up with negative constraints.

3 Preliminaries

Although the proposed methodology is suitable for any sufficiently expressive logical formalism, we focus our consideration on the OWL Web Ontology Language [13] as the currently most prominent expressive ontology language.[4]

[4] Note that RDF and RDFS, though certainly more widespread, do not allow for the specification of negative constraints. In fact, the only way to cause an inconsistency in RDFS – namely via XML-clashes – should be seen as a feature which was introduced accidentally rather than intentionally, cf. [7], Section 3.3.3.

Table 1. Syntax and semantics of role and class constructors in $\mathcal{SROIQ}$. Thereby a denotes an individual name, R an arbitrary role name and S a simple role name. C and D denote class expressions.

Name	Syntax	Semantics
inverse role	R^-	$\{\langle x,y\rangle \in \Delta^\mathcal{I} \times \Delta^\mathcal{I} \mid \langle y,x\rangle \in R^\mathcal{I}\}$
universal role	U	$\Delta^\mathcal{I} \times \Delta^\mathcal{I}$
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	$C \sqcup D$	$C^\mathcal{I} \cup D^\mathcal{I}$
nominals	$\{a\}$	$\{a^\mathcal{I}\}$
univ. restriction	$\forall R.C$	$\{x \in \Delta^\mathcal{I} \mid \langle x,y\rangle \in R^\mathcal{I}$ implies $y \in C^\mathcal{I}\}$
exist. restriction	$\exists R.C$	$\{x \in \Delta^\mathcal{I} \mid$ for some $y \in \Delta^\mathcal{I}$, $\langle x,y\rangle \in R^\mathcal{I}$ and $y \in C^\mathcal{I}\}$
Self class	$\exists S.\mathsf{Self}$	$\{x \in \Delta^\mathcal{I} \mid \langle x,x\rangle \in S^\mathcal{I}\}$
qualified number restriction	$\leqslant n\, S.C$	$\{x \in \Delta^\mathcal{I} \mid \#\{y \in \Delta^\mathcal{I} \mid \langle x,y\rangle \in S^\mathcal{I}$ and $y \in C^\mathcal{I}\} \leq n\}$
	$\geqslant n\, S.C$	$\{x \in \Delta^\mathcal{I} \mid \#\{y \in \Delta^\mathcal{I} \mid \langle x,y\rangle \in S^\mathcal{I}$ and $y \in C^\mathcal{I}\} \geq n\}$

The OWL DL version of the current OWL standard is based on the very expressive description logic $\mathcal{SROIQ}$ [8]. For a description of the relationship between OWL and the underlying description logics, the reader is referred to [7] or [16]. In this paper we will use description logic notation for its brevity. Thus, we briefly recap syntax and semantics of the description logic $\mathcal{SROIQ}$, although we will only actively work with a restricted sublanguage of it thereafter.

Let N_I, N_C, and N_R be finite, disjoint sets called *individual names*, *class names* and *role names* respectively,[5] with $N_R = \mathbf{R_s} \uplus \mathbf{R_n}$ called *simple* and *non-simple* roles, respectively. These atomic entities can be used to form complex classes and roles in the usual way (see Table 1). A *$\mathcal{SROIQ}$-knowledge base*[6] is a tuple $(\mathcal{T}, \mathcal{R}, \mathcal{A})$ where $\mathcal{T}$ is a $\mathcal{SROIQ}$-TBox, $\mathcal{R}$ is a regular $\mathcal{SROIQ}$-role hierarchy[7] and $\mathcal{A}$ is a $\mathcal{SROIQ}$-ABox containing axioms as presented in Table 2. The semantics of $\mathcal{SROIQ}$ 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 roles to subsets of $\Delta^\mathcal{I} \times \Delta^\mathcal{I}$. This mapping is extended to complex roles and classes as displayed in Table 1 and finally used to evaluate axioms (see Table 2). We say $\mathcal{I}$ satisfies a knowledge base $KB = (\mathcal{T}, \mathcal{R}, \mathcal{A})$ (or $\mathcal{I}$ is a model of KB, written: $\mathcal{I} \models KB$) if it satisfies all axioms of $\mathcal{T}$, $\mathcal{R}$, and $\mathcal{A}$. We say that a knowledge base KB *entails* an axiom α (written $KB \models \alpha$) if all models of KB are models of α. Finally, a knowledge base KB is satisfiable if it has a model and a class C is called satisfiable w.r.t. a knowledge

[5] Finiteness of the vocabulary is required for the further considerations. This does not impose a restriction since the vocabulary is not bounded and can be extended whenever this should be necessary.

[6] We use the terms knowledge base and ontology interchangeably.

[7] We assume the usual regularity assumption for $\mathcal{SROIQ}$, but omit it for space reasons.

Table 2. Syntax and semantics of $\mathcal{SROIQ}$ axioms

Axiom α	$\mathcal{I} \models \alpha$, if	
$R_1 \circ \cdots \circ R_n \sqsubseteq R$	$R_1^{\mathcal{I}} \circ \cdots \circ R_n^{\mathcal{I}} \subseteq R^{\mathcal{I}}$	RBox $\mathcal{R}$
$\mathsf{Dis}(S,T)$	$S^{\mathcal{I}} \cap T^{\mathcal{I}} = \emptyset$	
$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$	TBox $\mathcal{T}$
$C(a)$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$	ABox $\mathcal{A}$
$R(a,b)$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$	
$a \doteq b$	$a^{\mathcal{I}} = a^{\mathcal{I}}$	
$a \neq b$	$a^{\mathcal{I}} \neq b^{\mathcal{I}}$	

base KB if there is a model $\mathcal{I}$ of KB with $C^{\mathcal{I}} \neq \emptyset$. We also recap that C is satisfiable if and only if $KB \cup \{C(a)\}$ is satisfiable where a is a "fresh" individual not occurring in KB. Also, C is unsatisfiable if and only if $KB \models C \sqsubseteq \bot$.

4 Formal Foundations

We now define a subclass of OWL class expressions which we deem particularly intuitive to deal with from a cognitive perspective as they essentially represent (alternatives of) existing structures, while negations are only used at an elementary level.[8] To see that this choice is reasonable, note that humans would 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.

Definition 1. *Given sets N_C, N_R, N_I of atomic class names, atomic role names and individual names, respectively,* simple class expressions *are class expressions of one of the forms A, $\neg A$ (with $A \in N_C$), $\exists r.\top$, $\neg\exists r.\top$, $\exists r^-.\top$, $\neg\exists r^-.\top$ (for $r \in N_R$), $\{o\}$, $\neg\{o\}$ (for $o \in N_I$).*

Next, 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 $r \in N_R$ 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 the symbol X occurring exactly once in the place of an unnegated class name.*

In words, $\mathcal{CI}$ class expressions allow for the description of situations: existing objects, their interrelations and their properties (in terms of being (non)-members of atomic classes, (not) participating in a relationship, or (not) being

[8] In fact, the navigation paradigm for building such class expression will be such that it even discourages the use of this simple form of negation.

identical to a named individual). Thereby, the structure of the axioms enforces that only tree-like relationships can be described. Moreover, the use of disjunction allows for specifying alternatives for parts of the situation descriptions.

Pointed class expressions are used to put a *focus* on a subexpression of a class expression. This focus will serve as a marker to indicate a point in the expression where new subexpressions can be attached. Consequently, given a pointed $\mathcal{CI}$ class expression $C(X)$ and a $\mathcal{CI}$ class expression D, we write $C(D)$ for the class expression $C(X)[D/X]$ obtained by replacing the occurrence of X in $C(X)$ by D. The following proposition is an easy consequence of the observation that by construction, X occurs in a position with positive polarity.

Proposition 1. *Let KB be a knowledge base, let D and D' be arbitrary class expressions and let $C(X)$ be a pointed $\mathcal{CI}$ class expression. Then $KB \models D \sqsubseteq D'$ implies $KB \models C(D) \sqsubseteq C(D')$.*

Definition 2. *Given a knowledge base KB and a pointed $\mathcal{CI}$ class expression $C(X)$, we call $C(X)$ satisfiable w.r.t. KB, if $C(\top)$ is satisfiable w.r.t. KB. We further define the* possible adjuncts *of $C(X)$ (denoted by $\mathsf{poss}_{KB}(C(X))$) as all simple class expressions D for which $C(D)$ is satisfiable w.r.t. KB. Moreover, we define the* necessary adjuncts *of $C(X)$ (denoted by $\mathsf{nec}_{KB}(C(X))$) as the set of all simple class expressions D for which $C(\neg D)$ is unsatisfiable w.r.t. KB.*

Example 1. Let KB be a knowledge base containing just the following two axioms: (A1) $\exists \mathtt{colonyOf}^{-}.\top \sqcap \mathtt{EUCountry} \sqsubseteq \bot$ stating that EU countries must not have colonies and the axiom (A2) $\exists \mathtt{colonyOf}^{-}.\top \sqsubseteq \mathtt{Country}$ expressing that only countries may have colonies. Then, considering the pointed class expression $\mathtt{Country} \sqcap \exists \mathtt{colonyOf}.X$ has `Country` as a necessary adjunct since (A2) would render $\mathtt{Country} \sqcap \exists \mathtt{colonyOf}.\neg \mathtt{Country}$ unsatisfiable. On the other hand, `EUCountry` is *not* a possible adjunct, since $\mathtt{Country} \sqcap \exists \mathtt{colonyOf}.\mathtt{EUCountry}$ is not satisfiable.

Clearly, the sets of possible and necessary adjuncts of a pointed class expression provide useful information on how the expression can be reasonably extended and what extending adjuncts would be implied anyway, both taking the provided knowledge base into account. Still, in specific cases, with disjunctive information being involved, $\mathsf{poss}_{KB}(C(X))$ might not quite capture the needed information, as illustrated by the following example.

Example 2. Considering the knowledge base KB introduced above, the pointed class expression $\mathtt{EUCitizen} \sqcup \exists \mathtt{livesIn}.(\mathtt{EUCountry} \sqcup \exists \mathtt{colonyOf}.X)$ would allow for the class `EUCountry` as possible adjunct, as the class $\mathtt{EUCitizen} \sqcup \exists \mathtt{livesIn}.(\mathtt{EUCountry} \sqcup \exists \mathtt{colonyOf}.\mathtt{EUCountry})$ is still satisfiable thanks to either of the disjuncts `EUCitizen` and `EUCountry`.

To exclude such unwanted cases, we introduce the notion of balancedness (Definition 4) as a desired property of class expressions. Intuitively, a class expression is balanced, if all alternatives described by unions can possibly occur. Toward the formal definition, we first have to introduce the notion of prunings

(Definition 3). By pruning a pointed class expression, we specialize it by removing disjunctive side branches, thus enforcing that the disjunctive branch in which X is located must be "realized".

Definition 3. *The pruning of a pointed $\mathcal{CI}$ class expression is obtained by applying the recursive function* prune *(we tacitly exploit commutativity of $\sqcap$ and $\sqcup$ to reduce cases):*

$$\begin{aligned}
\mathsf{prune}(X) &:= X \\
\mathsf{prune}(C(X) \sqcap D) &:= \mathsf{prune}(C(X)) \sqcap D \\
\mathsf{prune}(C(X) \sqcup D) &:= \mathsf{prune}(C(X)) \\
\mathsf{prune}(\exists r.C(X)) &:= \exists r.\mathsf{prune}(C(X)) \\
\mathsf{prune}(\exists r^-.C(X)) &:= \exists r^-.\mathsf{prune}(C(X))
\end{aligned}$$

Example 3. Continuing the above example, we obtain

$$\begin{aligned}
&\mathsf{prune}(\texttt{EUCitizen} \sqcup \exists \texttt{livesIn}.(\texttt{EUCountry} \sqcup \exists \texttt{colonyOf}.X)) \\
&\qquad = \exists \texttt{livesIn}.\exists \texttt{colonyOf}.X.
\end{aligned}$$

Definition 4. *Let KB be a knowledge base and let C be a $\mathcal{CI}$ class expression in which a union $D_1 \sqcup D_2$ occurs as a subexpression. Let $C'(X)$ be obtained from C by replacing this occurrence with X, such that $C = C'(D_1 \sqcup D_2)$. Then, we call the occurrence of $D_1 \sqcup D_2$ in C* balanced *if both $\mathsf{prune}(C'(X))[D_1/X]$ and $\mathsf{prune}(C'(X))[D_2/X]$ are satisfiable w.r.t. KB. Otherwise, we say the occurrence is* imbalanced *and call every D_i with unsatisfiable $\mathsf{prune}(C'(X))[D_i/X]$ a* fake disjunct*.*

A $\mathcal{CI}$ class expression C is called fully balanced *if it is satisfiable and all occurrences of union subexpressions are balanced. A pointed class expression $C(X)$ is called fully balanced if $C(\top)$ is fully balanced.*

Example 4. $\texttt{EUCitizen} \sqcup \exists \texttt{livesIn}.(\texttt{EUCountry} \sqcup \exists \texttt{colonyOf}.\texttt{EUCountry})$ can be found to be not fully balanced since it contains the imbalanced occurrence of $\texttt{EUCountry} \sqcup \exists \texttt{colonyOf}.\texttt{EUCountry}$ with $\exists \texttt{colonyOf}.\texttt{EUCountry}$ being the fake disjunct since $\mathsf{prune}(\texttt{EUCitizen} \sqcup \exists \texttt{livesIn}.X)[\exists \texttt{colonyOf}.\texttt{EUCountry}/X] = (\exists \texttt{livesIn}.X)[\exists \texttt{colonyOf}.\texttt{EUCountry}/X] = \exists \texttt{livesIn}.\exists \texttt{colonyOf}.\texttt{EUCountry}$ is unsatisfiable w.r.t. KB (see above).

By definition, full balancedness of a class can be checked by a twofold class satisfiability test for each disjunctive subexpression, thus the number of necessary class satisfiability checks is linearly bounded by the size of the class expression.

It is rather easy to see that by restricting to fully balanced class expressions, we do not lose anything in terms of expressivity, since for any satisfiable class we find an equivalent one which is fully balanced by pruning away the fake disjuncts.

Proposition 2. *For any not fully balanced $\mathcal{CI}$ class expression C there is a $\mathcal{CI}$ class expression C' such that*

- *C' is fully balanced,*
- *$KB \models C \equiv C'$*
- *C' is obtained by the repeated replacement of imbalanced occurrences of unions by the respective non-fake disjunct.*

Example 5. Given KB from above, we find that the (not fully balanced) class expression $\texttt{EUCitizen} \sqcup \exists \texttt{livesIn}.(\texttt{EUCountry} \sqcup \exists \texttt{colonyOf}.\texttt{EUCountry})$ and the fully balanced class expression $\texttt{EUCitizen} \sqcup \exists \texttt{livesIn}.(\texttt{EUCountry})$ are equivalent w.r.t. KB.

These findings justify our suggestion to restrict the possible adjuncts for a pointed class expression to those which would not just maintain its satisfiability, but also its balancedness.

Definition 5. *Given a knowledge base KB and a pointed $\mathcal{CI}$ class expression $C(X)$, we define the* nice possible adjuncts *of $C(X)$ (denoted by $\mathsf{poss}^{\ddot\smile}_{KB}(C(X))$) as all simple class expressions D for which $C(D)$ is fully balanced w.r.t. KB.*

Example 6. Considering the knowledge base from above, we obtain $\texttt{EUCountry} \notin \mathsf{poss}^{\ddot\smile}_{KB}(\texttt{EUCitizen} \sqcup \exists \texttt{livesIn}.(\texttt{EUCountry} \sqcup \exists \texttt{colonyOf}.X))$ since inserting $\texttt{EUCountry}$ for X would result in a not fully balanced class.

5 Navigation

We now describe the navigation operations of our class exploration methodology on an abstract level as modifications of a pointed class expression $C(X)$.

(M) **Moving the focus.** For moving the focus, one picks an occurrence of a subclass D which is not in the scope of a negation. Then we obtain $C'(X, Y)$ from $C(X)$ by replacing the chosen occurrence of D by Y if $D = \top$ and by $D \sqcap Y$ otherwise. Thereafter, we obtain $C''(Y)$ from $C'(X, Y)$ by replacing $E \sqcap X$ by E if X occurs in such a conjunction, or otherwise replacing X by $\top$. Finally, we obtain the result $C_{\texttt{new}}(X)$ of this operation from $C''(Y)$ by substituting Y with X.

(D) **Deleting subexpression at focus.** This operation is applicable if X occurs in $C(X)$ inside a conjunction $E \sqcap X$. In this case, the result $C_{\texttt{new}}(X)$ is obtained by replacing $E \sqcap X$ by X.

(I) **Inserting a disjunction.** This operation is applicable if X occurs in $C(X)$ inside a conjunction $E \sqcap X$. In this case, the result $C_{\texttt{new}}(X)$ is obtained by replacing $E \sqcap X$ by $E \sqcup X$.

(E) **Extending the expression.** Pick a class expression $D \in \mathsf{poss}^{\ddot\smile}_{KB}(C(X))$ and obtain $C_{\texttt{new}}(X)$ by replacing X with $\exists r^{(-)}.X$ in case $D = \exists r^{(-)}.\top$ or otherwise with $D \sqcap X$.

We now provide two desirable properties, which justify the choice of the navigation steps introduced above. The not overly intricate but in places a bit verbose and tedious full proofs can be found in [3]. First, we show that in the course of navigation, only fully balanced classes can be obtained if one starts from a fully balanced pointed class expression.

Proposition 3. *Each of the described navigation steps* (M), (D), (I)*, and* (E) *results in a fully balanced pointed class expression, if it is applied to a fully balanced pointed class expression.*

The second proposition shows that the proposed navigation methodology is complete in the sense that we can construct all potentially "interesting" class expressions.

Proposition 4. *Each fully balanced pointed $\mathcal{CI}$ class expression can be constructed by a sequence of navigation steps starting from X.*

Summing up, our navigation paradigm is tuned in a way that favors construction of "meaningful" (in terms of satisfiability and balancedness) class descriptions but does not restrict expressivity otherwise.

6 The Possible World Explorer

We have developed a prototype of the Possible World Explorer (PEW[9] for short) that allows for both the exploration of possible worlds, and the assertion of negative axioms to eradicate possible worlds ("pew pew!"). On one hand, PEW is implemented on top of the OWL API[10] for handling ontologies, and on the HermiT reasoning engine [11] for checking the satisfiability of class expressions. On the other hand, PEW reuses the principles and user interface of SEWELIS[11] [2] for the interactive construction and display of class expressions and their possible adjuncts. The sources, executable, and screencasts of the system are available at `http://www.irisa.fr/LIS/softwares/pew/`.

Figure 1 shows a screenshot of PEW's user interface. It is composed of a toolbar (T) at the top, a class box (C) at the top left, an instance box (I) at the bottom left, and an adjunct box (A) on the right. The class box (C) displays the current pointed class expression $C(X)$, where the subexpression at the focus X is highlighted. The known instances of the class expression $C(\top)$ are listed in the instance box (I). The possible adjuncts of $C(X)$ are displayed in the adjunct box (A). Class names are displayed as a tree according to the underlying ontology's class hierarchy, and unqualified existential restrictions are displayed as a tree according to the property hierarchy. For each possible adjunct $D \in \mathsf{poss}^{\odot}_{KB}(C(X))$, both D and $\neg D$ are displayed, except for the necessary $D \in \mathsf{nec}_{KB}(C(X))$, for which only D is displayed but in a larger font. For the concrete syntax of the class expression and adjuncts, both DL notation and Manchester syntax are available. For better readability of complex expressions, we use indentation instead of brackets, and syntax highlighting (foreground color) to distinguish between class, role, and individual names.

From Figure 1, we can conclude a number of things about the pizza ontology [12]. From the class box, we conclude that a pizza *may* have no topping. From the emptiness of the instance box, we conclude that there is no known individual pizza without topping. From the adjunct box, we further conclude that such a

[9] We adopt the Semantic Web practice of flipping letters in acronyms.
[10] `http://owlapi.sourceforge.net/`
[11] `http://www.irisa.fr/LIS/softwares/sewelis/`
[12] `http://www.co-ode.org/ontologies/pizza/pizza.owl`

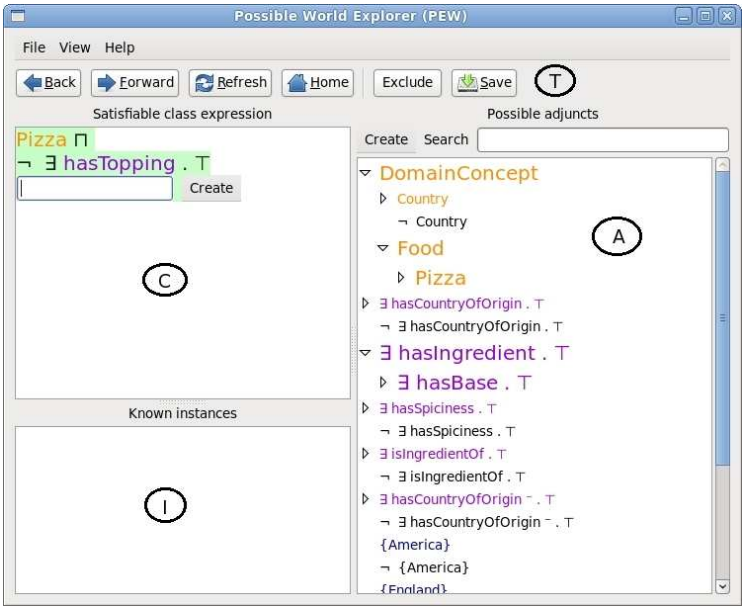

Fig. 1. Screenshot of the Possible World Explorer (PEW) showing that, according to the pizza ontology, a pizza can have no topping.

pizza *must* be some food, that it *must* have some base as an ingredient, but that it *may* also be a country, and that it *may* have no country of origin.

Navigation from one (pointed) class expression to another is entirely performed in an interactive way. Double-clicking an adjunct *extends* the class expression by inserting it at the focus. Alternatively, adjuncts can be found by auto-completion in the text fields (one at the focus in the class box, and another above the adjunct box). The focus can be *moved* simply by clicking on various parts of the class expression. The contextual menu of the class box (C) provides the additional navigation steps: *inserting a disjunction*, and *deleting* the subexpression at focus. The toolbar (T) provides navigation in the history of class expressions, as well as the update of the ontology. The button "Exclude" adds the axiom $C(\top) \sqsubseteq \bot$ to the ontology, in order to rule out models in which the current class expression has instances. Figure 1 displays a situation where this operation would make sense. To provide feedback about the update, the focus background color switches from green (satisfiable class) to red (unsatisfiable class). In the case where the update would make the ontology inconsistent, the button "Exclude" triggers an error message. Finally, the button "Save" saves the updated ontology.

The current implementation is rather naive, and optimization is left for future work. For information, we here give the nature and number of performed reasoning tasks (OWL API calls to reasoner methods). First, the hierarchy of simple class expressions has to be computed, which amounts to 1 call to `getInstances`

for the top class, 1 call to `getSubClasses` for each named class, and 1 call to `getSubObjectProperties` for each named role. Then, at each navigation step, the known instances of the class expression $C(X)$ are computed with 1 call to `getInstances`, and the possible adjuncts by checking the possibility and necessity of $C(D)$, for each positive simple class expression D. Checking possibility amounts to $1+2d$ call to `isSatisfiable`, where d is the number of unions in the class expression; and checking necessity amounts to 1 call to `isSatisfiable`.

7 An Example Scenario

In this section, we describe an example scenario of exploration and completion of the pizza ontology. This ontology has the advantage of being well-known, covering a large subset of OWL constructs, and being representative for OWL ontologies. While the pizza ontology is often referred to and was subject to a number of refinements, we found in our exploration a number of unexpected possible worlds, and hence missing axioms. The following scenario describes a non-exhaustive exploration, and illustrates various situations that may arise.

7.1 First Steps in the Ontology

After launching PEW on the pizza ontology, the initial pointed class expression is $C(X) = X$, the instance box displays the list of all named individuals, and the adjunct box displays all simple class expressions. The latter means that every simple class and its complement are satisfiable, which generally holds in ontologies. Without prior knowledge, the user can then discover that the ontology is about food (in particular pizzas, pizza bases and pizza toppings), countries, and spiciness. The possible roles are "has ingredient" (refined into "has base" and "has topping"), "has spiciness", "has country of origin", and their inverses. Only 5 named individuals exist, namely for countries.

7.2 Class Exploration

In order to better understand the possible interactions between classes and properties, the user decides to navigate to each named class to discover what an instance of that class can be. For example, by selecting the adjunct `Country`, the pointed class expression becomes $\texttt{Country} \sqcap X$ (see Figure 2). The instance box says there are 5 known countries, namely America, England, France, Germany, and Italy. Surprisingly, the adjunct box says that a country can be some food (possible adjunct `Food`), or not (possible adjunct $\neg\texttt{Food}$). This implies we can further select the adjunct `Food` to reach the satisfiable class expression $\texttt{Country} \sqcap \texttt{Food} \sqcap X$. Obviously, such an individual should not be possible, and we exclude this possibility by pushing the "Exclude" button, which has the effect of adding the axiom $\texttt{Country} \sqcap \texttt{Food} \sqsubseteq \bot$ to the ontology. This illustrates the claimed fact that even very basic negative constraints such as disjointness

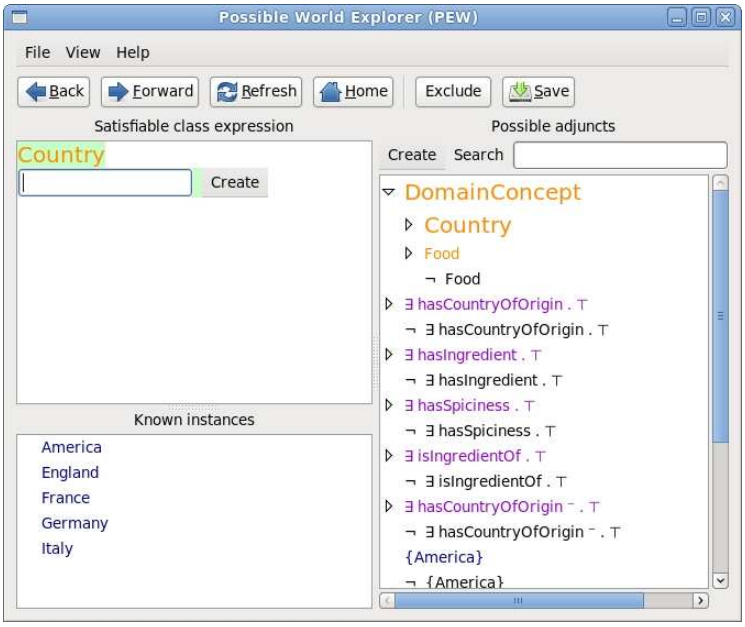

Fig. 2. A screenshot showing, among other things, that a country can be some food, and can have a country of origin.

axioms are often missing in ontologies. On the contrary, we found no missing positive axiom like subclass axioms.

Navigating back to $\texttt{Country} \sqcap X$, the user can verify that a country cannot anymore be an instance of another class. However, looking at possible roles, she discovers that a country can not only be the country of origin of something (adjunct $\exists \texttt{hasCountryOfOrigin}^{-}.\top$), which is fine, but can also have a country of origin (adjunct $\exists \texttt{hasCountryOfOrigin}.\top$), and a spiciness. Those two undesirable possibilities can be ruled out by selecting an unexpected adjunct, and asserting a negative axiom with the "Exclude" button, and by repeating this sequence on the other unexpected adjunct. Note that selecting the two adjuncts simultaneously, and then asserting an axiom would not be equivalent because this would only exclude countries that have both a country of origin *and* a spiciness. Yet, it is possible to use only one axiom provided a class union is used between the two unexpected adjuncts. At this stage, the user can see no more undesirable adjuncts for countries, and move to other named classes.

Looking at food ($C(X) = \texttt{Food} \sqcap X$), the only undesirable adjunct is that some food can be the country of origin of something, which the user excludes. Looking at pizzas, she decides to exclude the possibility for a pizza to be an ingredient. So far, we have only excluded positive possibilities (e.g., a pizza can be an ingredient), but it is also possible to exclude negative possibilities. From the adjunct box, the user discovers that, while a pizza must have some ingredient and

some base (the simple class $\neg\exists\texttt{hasBase}.\top$ is not a possible adjunct), it may have no topping (possible adjunct $\neg\exists\texttt{hasTopping}.\top$). This can be excluded simply by selecting the negative adjunct (instead of the positive one), and asserting the axiom $\texttt{Pizza} \sqcap \neg\exists\texttt{hasTopping}.\top \sqsubseteq \bot$ (see Figure 1), which is equivalent to $\texttt{Pizza} \sqsubseteq \exists\texttt{hasTopping}.\top$ (every pizza has a topping).

Finally, looking at spiciness (degrees), the user excludes the following possibilities: a spiciness that has a country of origin, a spiciness that is the country of origin of something, and a spiciness that has a spiciness (degree). After those exclusions, a spiciness can only be the spiciness of something. The class `Spiciness` has three subclasses: `Hot`, `Medium`, and `Mild`. Selecting any of those classes shows that no other class is possible simultaneously, which means that disjointness axioms have already been asserted between them.

7.3 Exploring Roles

When exploring roles, we investigate for each role what can be at their range and domain. Class exploration has covered axiom schemas $A \sqcap B \sqsubseteq \bot$, $A \sqcap \exists r.\top \sqsubseteq \bot$ and $A \sqcap \neg\exists r.\top \sqsubseteq \top$. Here, we will cover axiom schemas $\exists r.\top \sqcap \neg A \sqsubseteq \bot$ and $\exists r.\neg A \sqsubseteq \bot$, which correspond, respectively, to domain and range axioms.

Looking at things that have a country of origin (with focus on the range, i.e., $C(X) = \exists\texttt{hasCountryOfOrigin}.X$), the user finds that the country of origin may be not a country (adjunct $\neg\texttt{Country}$). This means that the range axiom for role `hasCountryOfOrigin` is missing. It can be added by selecting the undesirable adjunct, and asserting the axiom $\exists\texttt{hasCountryOfOrigin}.\neg\texttt{Country} \sqsubseteq \bot$ which is equivalent to $\top \sqsubseteq \forall\texttt{hasCountryOfOrigin}.\texttt{Country}$.

Inspecting things having a spiciness ($C(X) = \exists\texttt{hasSpiciness}.\top \sqcap X$) with focus at the domain, it appears that those things may not be food (adjunct $\neg\texttt{Food}$). This can be excluded by asserting the axiom $\exists\texttt{hasSpiciness}.\top \sqcap \neg\texttt{Food} \sqsubseteq \bot$, which is equivalent to $\exists\texttt{hasSpiciness}.\top \sqsubseteq \texttt{Food}$, and defines a domain axiom.

7.4 Further Exploration

In this section, we provide an additional example to show that our approach does not only apply to simple interactions between named classes and roles. The class `Pizza` has a number of subclasses that are generally defined through equivalent classes axioms as pizzas satisfying certain criteria. For example, there is the `VegetarianPizza` class, and obviously it should not be possible to find a vegetarian pizza that contains some meat or fish as an ingredient. Following the *advocatus diaboli* approach, this is exactly what we are going to try and find. Starting from a vegetarian pizza ($C(X) = \texttt{VegetarianPizza} \sqcap X$), we find that it may (and must) have some ingredient (adjunct $\exists\texttt{hasIngredient}.\top$). By selecting this adjunct, we reach the class expression $C(X) = \texttt{VegetarianPizza} \sqcap \exists\texttt{hasIngredient}.X$, with the focus on the ingredient. As possible ingredients, we find only food, subdivided into pizza base and pizza topping. Under the class `PizzaTopping`, we find that both subclasses `MeatTopping` and `FishTopping` are possible! It is even possible to reach a vegetarian pizza that has both meat

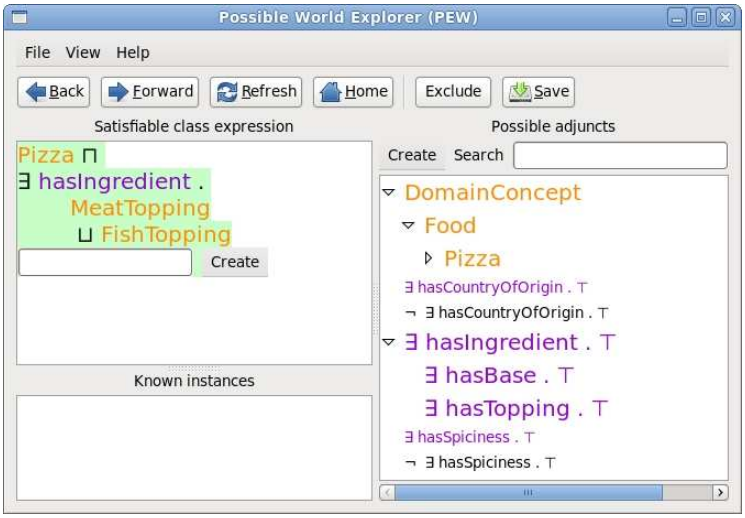

Fig. 3. A screenshot showing that a vegetarian pizza may contain some meat or fish.

and fish as ingredients. The two undesirable possibilities can be excluded at once by navigating to the satisfiable and balanced class `VegetarianPizza` ⊓ ∃`hasIngredient`.(`MeatTopping` ⊔ `FishTopping`) (see Figure 3), and pushing the Exclude button. Looking at the ontology, we do find that a vegetarian pizza is defined as a pizza that contains neither meat nor fish. The problem is that in the respective axiom, the role "has topping" was used instead of the more general "has ingredient". Obviously, a vegetarian pizza should have no meat or fish, no matter what part of the pizza contains it!

8 Conclusion and Future Work

We have proposed an intuitive methodology for adding negative constraints to OWL ontologies in an exploratory way. To this end, we devised and implemented an interaction paradigm for constructing intuitive satisfiable class expressions in an interactive way reminiscent of faceted browsing, which–if found to be absurd–can be turned unsatisfiable by adding a corresponding negative constraint to the underlying ontology.

Future work on this subject clearly includes scalability and usability investigations and improvements. For seamless navigation and editing, the underlying reasoning steps must be performed in near-realtime which poses some restriction on the computational intricacy of the considered ontology. In order to enlarge the scope of applicability of our method, we will optimize PEW in terms of minimizing OWL API calls.

On the usability side, next to thorough user studies, we will further enrich the tool with further functionality beyond mere exclusion of unwanted class expressions. Ultimately, we plan to provide PEW as a Protégé plugin.

Acknowledgements This work was performed in the course of a research visit of Sebastian Rudolph in Rennes supported by IRISA and University Rennes 1.

References

1. Baader, F., Ganter, B., Sertkaya, B., Sattler, U.: Completing description logic knowledge bases using formal concept analysis. In: Veloso, M.M. (ed.) IJCAI. pp. 230–235 (2007)
2. Ferré, S., Hermann, A.: Semantic search: Reconciling expressive querying and exploratory search. In: Aroyo, L., Welty, C. (eds.) Int. Semantic Web Conf. LNCS, vol. 7031, pp. 177–192. Springer (2011)
3. Ferré, S., Rudolph, S.: PEW! PEW! Tech. rep., Institut AIFB, KIT, Karlsruhe (April 2012), available via http://www.aifb.kit.edu/web/Techreport3024
4. Fleischhacker, D., Völker, J.: Inductive learning of disjointness axioms. In: Meersman, R., Dillon, T.S., Herrero, P., Kumar, A., Reichert, M., Qing, L., Ooi, B.C., Damiani, E., Schmidt, D.C., White, J., Hauswirth, M., Hitzler, P., Mohania, M.K. (eds.) OTM Conferences (2). LNCS, vol. 7045, pp. 680–697. Springer (2011)
5. Gómez-Pérez, A., Euzenat, J. (eds.): The Semantic Web: Research and Applications, Second European Semantic Web Conference, ESWC 2005, Heraklion, Crete, Greece, May 29 - June 1, 2005, Proceedings, LNCS, vol. 3532. Springer (2005)
6. Haase, P., Stojanovic, L.: Consistent evolution of OWL ontologies. In: Gómez-Pérez and Euzenat [5], pp. 182–197
7. Hitzler, P., Krötzsch, M., Rudolph, S.: Foundations of Semantic Web Technologies. Chapman & Hall/CRC (2009)
8. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible $\mathcal{SROIQ}$. In: Doherty, P., Mylopoulos, J., Welty, C.A. (eds.) Proc. 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'06). pp. 57–67. AAAI Press (2006)
9. Meilicke, C., Stuckenschmidt, H., Tamilin, A.: Repairing ontology mappings. In: AAAI. pp. 1408–1413. AAAI Press (2007)
10. Meilicke, C., Völker, J., Stuckenschmidt, H.: Learning disjointness for debugging mappings between lightweight ontologies. In: Gangemi, A., Euzenat, J. (eds.) EKAW. LNCS, vol. 5268, pp. 93–108. Springer (2008)
11. Motik, B., Shearer, R., Horrocks, I.: Hypertableau reasoning for description logics. J. of Artificial Intelligence Research 36, 165–228 (2009)
12. Nikitina, N., Rudolph, S., Glimm, B.: Interactive ontology revision. Web Semantics: Science, Services and Agents on the World Wide Web 12-13(0), 118–130 (2012)
13. OWL Working Group, W.: OWL 2 Web Ontology Language: Document Overview. W3C Recommendation (27 October 2009), available at http://www.w3.org/TR/owl2-overview/
14. Rudolph, S.: Exploring relational structures via FLE. In: Wolff, K.E., Pfeiffer, H.D., Delugach, H.S. (eds.) ICCS. LNCS, vol. 3127, pp. 196–212. Springer (2004)
15. Rudolph, S.: Acquiring generalized domain-range restrictions. In: Medina, R., Obiedkov, S.A. (eds.) ICFCA. LNCS, vol. 4933, pp. 32–45. Springer (2008)
16. Rudolph, S.: Foundations of description logics. In: Polleres, A., d'Amato, C., Arenas, M., Handschuh, S., Kroner, P., Ossowski, S., Patel-Schneider, P.F. (eds.) Reasoning Web. LNCS, vol. 6848, pp. 76–136. Springer (2011)
17. Schlobach, S.: Debugging and semantic clarification by pinpointing. In: Gómez-Pérez and Euzenat [5], pp. 226–240
18. Völker, J., Vrandecic, D., Sure, Y., Hotho, A.: Learning disjointness. In: Franconi, E., Kifer, M., May, W. (eds.) ESWC. LNCS, vol. 4519, pp. 175–189. Springer (2007)