HAL
archives-ouvertes.fr

Conception interactive d'ontologies par élimination de mondes possibles

Sébastien Ferré

▶ **To cite this version:**

Sébastien Ferré. Conception interactive d'ontologies par élimination de mondes possibles. IC2015, Jun 2015, Rennes, France. collection AFIA, 2015, collection AFIA. <hal-01165493>

HAL Id: hal-01165493
https://hal.inria.fr/hal-01165493

Submitted on 19 Jun 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Conception interactive d'ontologies par élimination de mondes possibles

Sébastien Ferré

IRISA/Université de Rennes 1
Campus de Beaulieu, 35042 Rennes cedex
ferre@irisa.fr

Résumé : La conception d'ontologies constitue souvent un frein à l'adoption des techniques de l'ingénierie des connaissances et du Web sémantique. Une raison est bien sûr l'emploi de formalismes et des concepts logiques qui y sont associés. Une autre raison qui nous semble plus profonde est le fossé entre syntaxe et sémantique, c'est-à-dire entre la forme de surface de l'ontologie (axiomes) et ce qu'elle rend nécessaire/possible/impossible (modèles). Ce fossé entraîne des divergences entre l'intention du concepteur et sa modélisation qui se manifestent par des inférences inattendues, voire des incohérences. Nous proposons une nouvelle approche de conception d'ontologies fondée sur l'exploration et l'élimination interactive de "mondes possibles" (modèles). Elle réduit le fossé syntaxe/sémantique en interdisant par construction la production d'incohérence, et en montrant en permanence au concepteur ce qui peut être inféré ou non. Un prototype, PEW (Possible World Explorer), permet d'expérimenter cette approche et de la comparer à d'autres éditeurs d'ontologies.

Mots-clés : Web sémantique, ontologies, OWL, conception, fossé syntaxe/sémantique, approche interactive.

1 Introduction

La conception d'ontologies est généralement une étape essentielle dans l'application des techniques d'ingénierie des connaissances et du Web sémantique. Les méthodologies existantes distinguent en général les phases de *conceptualization* et de *formalisation* dans un langage formel. On s'intéresse ici à la phase de formalisation qui pose un certain nombre de difficultés, en particulier pour les débutants mais pas seulement. Certaines difficultés sont liées à la manipulation d'un langage formel tel que OWL, et des outils tels que Protégé visent à réduire ce type de difficultés. D'autres difficultés sont liées aux différences qui peuvent survenir entre l'intension du concepteur et ce qu'exprime vraiment l'ontologie (Rector *et al.*, 2004; Corman, 2013). Par exemple, "ne mange que des légumes" n'implique pas "mange des légumes". Ou encore, savoir que "X est une femme" ne permet pas de déduire que "X n'est pas un homme" si on n'a pas explicitement dit que "hommes et femmes forment des classes séparées". Les contraintes négatives, telles que les séparations de classes ou les inégalités entre individus, sont souvent omises tellement elles semblent aller de soi. Leur omission est difficile à détecter car elle ne se manifeste pas par des inférences erronées, mais par des absences d'inférence.

Les éditeurs d'ontologie tels que Protégé (Noy *et al.*, 2001) favorisent l'expression de contraintes positives, c'est-à-dire d'axiomes permettant l'inférence de faits positifs : ex., hiérarchie de classes, domaines et co-domaines des propriétés. L'utilisateur a donc avant tout une vision syntaxique de son ontologie et se trouve très peu en contact avec sa sémantique, c'est-à-dire ce qu'elle rend possible ou non comme situation. Il est possible de faire appel à un raisonneur pour tester la cohérence de l'ontologie ou la satisfiabilité d'une classe. Un certain nombre d'outils existent aussi pour détecter des erreurs courantes et compléter les ontologies de façon systématique (Meilicke *et al.*, 2008; Poveda-Villalón *et al.*, 2012; Corman, 2013). Cependant,

ces approches ne sont pas constructives, mais correctives. De plus, elles sont généralement limitées aux axiomes de séparation, la forme la plus simple de contrainte négative. Dans un papier précédent (Ferré & Rudolph, 2012), nous avons montré des erreurs et omissions importantes dans l'ontologie des pizzas [1], laquelle sert pourtant de modèle et de support pédagogique depuis longtemps. Par exemple, les classes `Food` et `Country` ne sont pas séparées, et il s'avère qu'une pizza végétarienne peut bien contenir de la viande ou du poisson comme ingrédient.

Nous proposons une nouvelle approche de la formalisation d'ontologies qui est centrée sur la sémantique plutôt que sur la syntaxe. Plutôt que de voir une ontologie comme un ensemble d'axiomes, nous proposons de la voir comme un ensemble de modèles, c'est-à-dire comme l'ensemble des interprétations autorisées par l'ontologie. Et plutôt que de voir la construction d'une ontologie comme l'ajout d'axiomes, nous proposons de la voir comme l'élimination de modèles. Chaque élimination d'un sous-ensemble de modèles produit un axiome et on obtient bien au final un ensemble d'axiomes, mais celui-ci n'est que le résultat du processus de conception, pas le moyen. Le principal avantage de cette approche est de permettre au concepteur de travailler au niveau des instances – les mondes possibles – comme pour le peuplement d'une ontologie (connaissances particulières), mais en définissant néanmoins le niveau terminologique de l'ontologie (connaissances générales).

La structure de l'article suit le cycle "exploration-élimination" de notre approche. La section 2 présente tout d'abord l'exploration des mondes possibles d'une ontologie. La section 3 explique ensuite la conception d'ontologie par l'élimination de mondes possibles trouvés lors de l'exploration. Nous illustrons ces deux phases avec le prototype PEW [2] et l'ontologie des pizzas, et rapportons de premières expériences en Section 4. Cet article s'appuie sur les résultats théoriques d'un travail précédent (Ferré & Rudolph, 2012) et en étend l'application à la conception d'ontologies.

2 Exploration de mondes possibles

Nous commençons par décrire ici une méthode d'exploration sûre et complète des mondes possibles d'une ontologie OWL. Aucune hypothèse n'est faite sur cette ontologie. Elle peut contenir des instances ou non. Elle peut se limiter à des taxonomies ou bien contenir des axiomes complexes. Pour des raison de concision, nous adoptons la notation des logiques de descriptions (Baader *et al.*, 2003) pour les axiomes et expressions de classe. PEW offre le choix entre cette notation et la notation de Manchester.

À tout moment, une vue partielle sur les mondes possibles est présentée au concepteur d'ontologie. Cette vue comprend trois parties interdépendantes (voir la figure 1 montrant une capture d'écran du prototype PEW) : une expression de classe satisfiable avec focus, une liste d'instances connues et une liste d'incréments possibles (*adjuncts*). L'expression de classe combine des classes, des propriétés et des individus et décrit une situation rendue possible par l'ontologie. Dans la figure 1, l'expression de classe décrit la situation où une pizza n'a pas de *topping*. Le focus désigne une sous-expression de cette expression de classe (surlignée en vert dans PEW) et sert à "mettre le focus" sur une entité de la situation décrite. Dans l'exemple, le focus est mis sur la pizza. Les instances connues sont tous les individus de l'ontologie qui sont

1. `http://protege.stanford.edu/ontologies/pizza/pizza.owl`
2. `http://www.irisa.fr/LIS/softwares/pew`

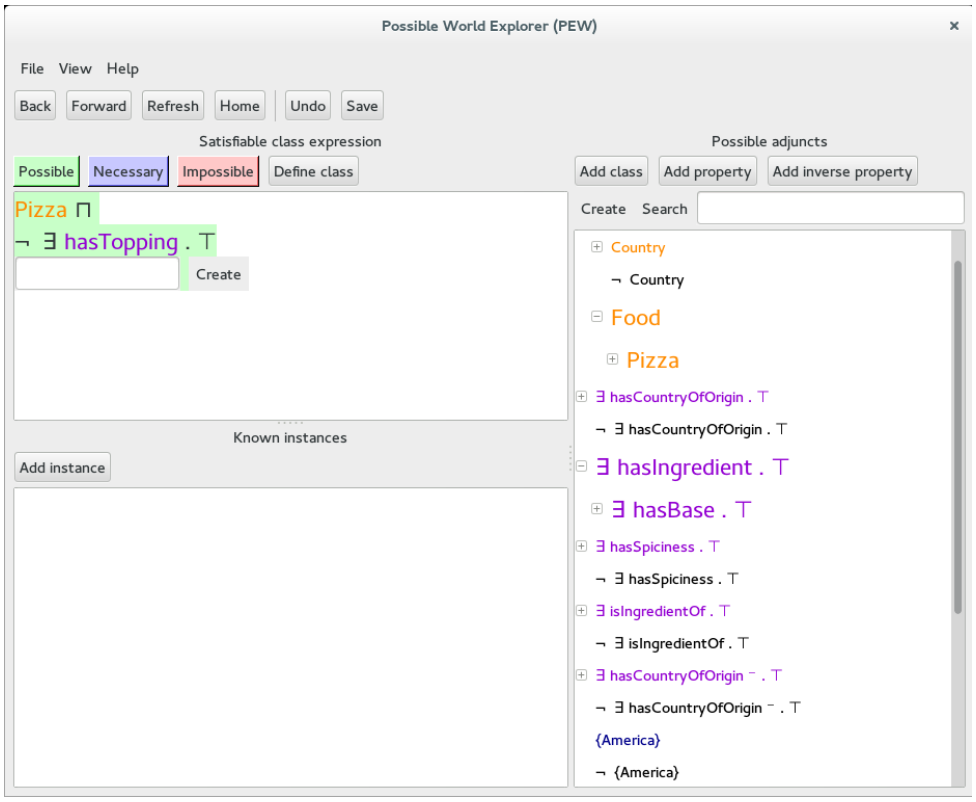

FIGURE 1 – Capture d'écran de PEW explorant les pizzas sans *topping*.

nécessairement des instances de l'expression de classe. Dans l'exemple de la figure 1, il n'y en a pas. Enfin, les incréments possibles sont toutes les classes simples qui sont "compatibles" avec le focus dans l'expression de classe. Une classe simple est "compatible" si son insertion au focus préserve la satisfiabilité de l'expression de classe. Une classe simple a une forme parmi : A, $\exists r.\top$, $\exists r^-.\top$, $\{o\}$, et leurs compléments. Si une classe simple est possible mais pas son complément, elle est dite *nécessaire*. Dans l'exemple de la figure 1, on voit qu'une pizza sans *topping* a nécessairement une base ($\exists hasBase.\top$), et que, de façon surprenante, elle peut être un pays ($Country$).

Un grand nombre de situations peuvent être décrites par une expression de classe. Le langage couvert comprend les classes atomiques (A), les restrictions existentielles qualifiées ($\exists r.C$), les propriétés inverses (r^-), la classe *top* ($\top$), les intersections ($C \sqcap D$), les unions ($C \sqcup D$), les compléments ($\neg C$) et les classes nominales ($\{o\}$). Les restrictions universelles sont indirectement couvertes par combinaison de restrictions existentielles et de compléments ($\forall r.C \equiv \neg\exists r.\neg C$). Les classes atomiques et nominales, ainsi que les restrictions existentielles, sont insérées dans l'expression de classe par sélection d'incréments. Les sélections successives forment des conjonctions. Les restrictions existentielles $\exists r.\top$ peuvent être qualifiées en mettant le focus sur le *top* $\top$, puis en sélectionnant des incréments. Les unions et les compléments sont insérés au focus via le menu contextuel. Si le focus est une sous-expression de classe C, l'insertion d'une union la remplace par l'expression $C \sqcup \top$, où $\top$ initialise une situation alternative à C. Celle-ci peut alors être affinée par insertion d'incréments. Si le focus est une sous-expression C,

axiome	ensemble équivalent d'axiomes de la forme $C \sqsubseteq \bot$
$C \sqsubseteq D$	$C \sqcap \neg D \sqsubseteq \bot$
$C(a)$	$\{a\} \sqcap \neg C \sqsubseteq \bot$
$r(a,b)$	$\{a\} \sqcap \neg \exists r.\{b\} \sqsubseteq \bot$, $\{b\} \sqcap \neg \exists r^{-}.\{a\} \sqsubseteq \bot$
$a = b$	$\{a\} \sqcap \neg \{b\} \sqsubseteq \bot$, $\{b\} \sqcap \neg \{a\} \sqsubseteq \bot$
$a \neq b$	$\{a\} \sqcap \{b\} \sqsubseteq \bot$

TABLE 1 – Équivalences pour les principaux axiomes de logiques de description.

l'insertion d'un complément la remplace par l'expression $\neg C$.

Nous avons démontré dans un papier précédent (Ferré & Rudolph, 2012) la *sûreté* et la *complétude* du processus de construction d'expressions de classe, sauf pour l'insertion de compléments que nous discutons ci-après. La sûreté implique que toute expression de classe pouvant être construite est satisfiable, et la complétude implique que toute expression de classe satisfiable peut être construite en un nombre fini d'étapes. Pour être sûre, l'insertion d'un complément dans une expression de classe $C \sqcap \underline{D}$ (focus sur D) n'est permise que si l'expression $C \sqcap \neg D$ est satisfiable. Dans le cas contraire, D est nécessairement vrai quand C est vrai (C est subsumé par D), et cette information est reflétée dans PEW par un focus de couleur bleu (signifiant "nécessaire") plutôt que vert (signifiant "possible").

La construction interactive d'expressions de classes satisfiables permet une exploration des mondes possibles d'une ontologie. Le simple fait de pouvoir ou non construire une expression de classe renseigne déjà sur les situations rendues possibles ou non par l'ontologie. Dans l'exemple de la figure 1, on découvre par exemple que l'ontologie des pizzas autorise une pizza à ne pas avoir de *topping*. Ensuite, la liste des instances connues indique directement au concepteur si la classe courante est habitée. Enfin, la liste des incréments indique pour toutes les classes simples si elles sont nécessaires, possibles ou impossibles. Dans l'exemple, on découvre qu'une pizza doit avoir une base, ce qui est correct, mais peut également être un pays, ce qui est absurde. Ces incréments correspondent à autant de réponses à des questions que le concepteur n'a même pas eu à poser. Ils offrent un *feedback* précieux sur la sémantique de l'ontologie explorée.

3 Conception par élimination de mondes possibles

Dans l'approche proposée dans cet article, la conception d'une ontologie fonctionne essentiellement par l'élimination de mondes possibles, mais également par l'augmentation de la signature de l'ontologie, c'est-à-dire par la création de nouvelles entités OWL : classes, propriétés et individus. La création de nouvelles entités OWL peut se faire soit à la volée, lors de la construction d'une expression de classe (bouton `Create` dans la figure 1), soit en les ajoutant à la liste des incréments possibles (voir boutons `Add class`, `Add property` et `Add inverse property`). Ces créations ne produisent aucun axiome, mais permettent le démarrage à froid dans la conception d'ontologie. Par exemple, pour l'ontologie de pizzas, on peut commencer par créer les classes $Pizza$ et $Country$, et la propriété $ingredient$. À ce stade, tout est possible : une pizza peut être un pays, un pays peut être un ingrédient de pizza, etc.

L'élimination de mondes possibles s'effectue en déclarant que la situation décrite par l'expression de classe courante C est impossible (voir bouton `Impossible` dans la figure 1). Vi-

suellement, dans PEW, la couleur du focus passe alors du vert au rouge. Rendre une classe impossible revient à la rendre vide de tout individu, et donc à en faire une sous-classe de la classe *bottom* $\bot$. Cette élimination produit donc l'axiome $C \sqsubseteq \bot$. Dans l'ontologie de pizza, on peut par exemple rendre impossible les expressions de classes suivantes : $Pizza \sqcap Country$ ("les pizzas ne sont pas des pays et réciproquement"), $Country \sqcap (\exists ingredient.\top \sqcup \exists ingredient^{-}.\top)$ ("les pays ne sont pas des ingrédients et n'ont pas d'ingrédients"), $Pizza \sqcap \neg\exists ingredient.\top$ ("les pizzas ont nécessairement au moins un ingrédient"). Grâce aux compléments de classes et aux classes nominales, la plupart des axiomes des logiques de description, et donc de OWL, peuvent être reformulés sous cette forme par un ou deux axiomes. La table 1 donne ces équivalences. Les axiomes qui ne peuvent pas être reformulés ainsi sont les axiomes portant sur les propriétés : subsomption, réflexivité, transitivité, etc.

Le prototype PEW offre un certain nombre de raccourcis pour exprimer des impossibilités. Les motivations de ces raccourcis sont : (a) une plus grande efficacité dans l'interaction et la production d'axiomes, et (b) l'évitement de l'emploi de compléments pour les axiomes positifs simples de type $A \sqsubseteq B$ et $C(a)$. Chaque raccourci est associé à une des trois zones de l'interface. Au-dessus de l'expression de classe, le bouton `Necessary` permet de rendre la sous-expression du focus nécessaire par rapport au reste de l'expression. La couleur du focus passe alors du vert au bleu. Ce raccourci produit des axiomes de la forme $C \sqcap \neg D \sqsubseteq \bot$, équivalente à $C \sqsubseteq D$, où D est la sous-expression au focus. Par exemple, partant de l'expression $AmericanPizza \sqcap \underline{\exists origin.\{America\}}$, on produit l'axiome $AmericanPizza \sqsubseteq \exists origin.\{America\}$ ("Toute pizza American est originaire d'Amérique"). En partant d'un autre focus, $AmericanPizza \sqcap \exists origin.\underline{\{America\}}$, on produit l'axiome $\exists origin^{-}.AmericanPizza \sqsubseteq \{America\}$ ("La seule origine des pizza American est l'Amérique"). Le raccourci `Define class` permet de créer une nouvelle classe A comme équivalente à l'expression de classe courante C. Il produit donc l'axiome $A \equiv C$, équivalent aux deux axiomes $A \sqsubseteq C$ et $C \sqsubseteq A$. On peut ainsi définir la classe $VegetarianPizza$ comme équivalente à l'expression $Pizza \sqcap \neg\exists ingredient.(Meat \sqcup Fish)$.

Au-dessus de la liste d'instance, le bouton `Add instance` permet de créer un nouvel individu a qui soit une instance de l'expression de classe courante C. Cela produit l'axiome $\{a\} \sqcap \neg C \sqsubseteq \bot$, équivalent à $C(a)$. Dans l'arbre d'incréments possible, le menu contextuel offre plusieurs raccourcis s'appliquant à une sélection de un ou plusieurs incréments. Tout d'abord, la commande `Add subclass` appliquée à une classe A permet de créer une nouvelle classe B et d'en faire une sous-classe de A en produisant l'axiome $B \sqsubseteq A$. Ce type d'axiome est en effet très commun. La commande `All impossible` (resp. `All necessary`) permet de rendre chaque incrément sélectionné impossible (resp. nécessaire) par rapport à l'expression de classe courante. Elle a le même effet que d'insérer successivement chaque incrément puis de presser le bouton `Impossible` (resp. `Necessary`). Ces commandes permettent de définir très rapidement le domaine et le co-domaine des propriétés, ainsi que les propriétés pouvant ou devant s'appliquer aux classes : par exemple, "les pays ne sont pas des ingrédients et n'ont pas d'ingrédients", et "les pizzas doivent avoir un ingrédient". Enfin, la commande `All disjoint` permet d'établir la séparation deux-à-deux d'un ensemble de classes simples. Elle permet donc aussi bien de générer les axiomes de la forme $A \sqcap B \sqsubseteq \bot$ (ex., $Pizza$ et $Country$) que de la forme $a \neq b$ (ex., $America$ et $France$). Un ensemble de n classes simples génère $n(n-1)/2$ axiomes et cette commande offre donc un raccourci précieux. À cause de leur combinatoire, ces axiomes sont souvent omis alors qu'ils sont cruciaux pour l'inférence.

4 Premières expériences

Nous avons réalisé de premières expériences qui se sont révélées encourageantes. Nous avons recréé le schéma de l'ontologie de pizzas dans l'ordre suivant : création de la hiérarchie de classes et séparation des classes sœurs entre elles ; pour chaque classe, création des instances connues (par ex. pays) et séparation entre ces instances ; création des propriétés et définition de leur domaines et co-domaines ; nécessité et impossibilité de certaines propriétés pour chaque classe. À ce stade, nous avons créé en quelques clics toutes les contraintes négatives pertinentes, contrairement à l'ontologie originale. Par contre, les axiomes de propriétés ne sont pas couverts (par ex. transitivité de $hasTopping$). Puis nous avons décrit différentes sortes de pizzas, telle que l'*American*. Il est possible de spécifier la liste de *toppings* d'une pizza sans utiliser de restriction universelle, ni de complément, mais l'utilisation d'une union est nécessaire pour exclure tout autre *topping*. En effet, un axiome de la forme $X \sqsubseteq \forall r.(A \sqcup B)$ peut être obtenu en rendant nécessaire le focus dans l'expression de class $X \sqcap \exists r.(\underline{A \sqcup B})$.

5 Conclusion et perspectives

La conception d'ontologie par élimination de mondes possibles présente l'avantage de montrer la réalité sémantique d'une ontologie et ainsi d'assister le concepteur dans la définition la plus complète possible de son ontologie. Cependant, l'affichage de tous les possibles peut être déroutant pour le concepteur car ils sont souvent bien plus nombreux que ce qui est attendu intuitivement et car il peut être fastidieux de les éliminer tous. Il sera nécessaire de faire des évaluations utilisateurs pour valider notre approche. Notons qu'une implémentation sous forme de plugin Protégé permettrait d'entrelacer librement cette approche avec l'approche classique.

Références

F. BAADER, D. CALVANESE, D. L. MCGUINNESS, D. NARDI & P. F. PATEL-SCHNEIDER, Eds. (2003). *The Description Logic Handbook : Theory, Implementation, and Applications*. Cambridge University Press.

CORMAN J. (2013). Explorer les théorèmes d'une TBox. In *Journées francophones d'Ingénierie des Connaissances*.

FERRÉ S. & RUDOLPH S. (2012). Advocatus diaboli - exploratory enrichment of ontologies with negative constraints. In A. TEN TEIJE ET AL., Ed., *Int. Conf. Knowledge Engineering and Knowledge Management (EKAW)*, LNAI 7603, p. 42–56 : Springer.

MEILICKE C., VÖLKER J. & STUCKENSCHMIDT H. (2008). Learning disjointness for debugging mappings between lightweight ontologies. In A. GANGEMI & J. EUZENAT, Eds., *EKAW*, volume 5268 of *LNCS*, p. 93–108 : Springer.

NOY N., SINTEK M., DECKER S., CRUBEZY M., FERGERSON R. & MUSEN M. (2001). Creating semantic web contents with Protege-2000. *Intelligent Systems, IEEE*, **16**(2), 60–71.

POVEDA-VILLALÓN M., SUÁREZ-FIGUEROA M. & GÓMEZ-PÉREZ A. (2012). Validating ontologies with OOPS ! In *Knowledge Engineering and Knowledge Management (EKAW)*, p. 267–281. Springer.

RECTOR A., DRUMMOND N., HORRIDGE M., ROGERS J., KNUBLAUCH H., STEVENS R., WANG H. & WROE C. (2004). OWL pizzas : Practical experience of teaching OWL-DL : Common errors & common patterns. In *Engineering Knowledge in the Age of the Semantic Web*, p. 63–81. Springer.