Detecting and Revising Flaws in OWL Object Property Expressions

C. Maria Keet

School of Mathematics, Statistics, and Computer Science, University of KwaZulu-Natal, and UKZN/CSIR-Meraka Centre for Artificial Intelligence Research, South Africa, keet@ukzn.ac.za

Abstract. OWL 2 DL is a very expressive language and has many features for declaring complex object property expressions. Standard reasoning services for OWL ontologies assume the axioms in the 'object property box' to be correct and according to the ontologist's intention. However, the more one can do, the higher the chance modelling flaws are introduced; hence, an unexpected or undesired classification or inconsistency may actually be due to a mistake in the object property box, not the class axioms. We identify the types of flaws that can occur in the object property box and propose corresponding compatibility services, *SubProS* and *ProChainS*, that check for meaningful property hierarchies and property chaining and propose how to revise a flaw. *SubProS* and *ProChainS* were evaluated with several ontologies, demonstrating they indeed do serve to isolate flaws and can propose useful corrections.

1 Introduction

There is considerable ongoing ontology development effort in various subject domains, such as the life sciences, medicine, e-learning, and the enterprise. New ontologies tend to be developed with the most recent W3C-standardised ontology language, OWL 2 [1]. OWL 2 DL is based on the Description Logics (DL) language $\mathcal{SROIQ}$, which, thanks to ontology engineers' requests for more features for object properties, now allows for object sub-properties, (inverse) functional, disjointness, equivalence, cardinality, (ir)reflexivity, (a)symmetry, transitivity, and role chaining. There are a few syntactic constraints on their usage, but still a lot is possible to declare, which also means there is now even more room to make mistakes with respect to the ontologist's intention in addition to those noted for modelling with OWL 1 [2–4]. For instance,

(i) Domain and range flaws; e.g. (simplified), $\mathsf{hasParent} \sqsubseteq \mathsf{hasMother}$ instead of $\mathsf{hasMother} \sqsubseteq \mathsf{hasParent}$ in accordance with their domain and range restrictions, or declaring a domain/range to be an intersection of disjoint classes;
(ii) Property characteristics flaws: e.g., the `family-tree`[1] has $\mathsf{hasGrandFather} \sqsubseteq \mathsf{hasAncestor}$ and $\mathsf{Trans}(\mathsf{hasAncestor})$ so that transitivity unintentionally is passed down the property hierarchy ($\mathsf{hasGrandFather}$ is intransitive, which cannot be asserted in OWL);

[1] `http://www.co-ode.org/roberts/family-tree.owl`; last accessed 12-3-2012.

(iii) Property chain issues; e.g., $\mathsf{hasPart} \circ \mathsf{hasParticipant} \sqsubseteq \mathsf{hasParticipant}$ in the pharmacogenomics ontology [5] that forces the classes in class expressions using these properties (DrugTreatment and DrugGeneInteraction) to be either processes due to the domain of hasParticipant, or they will be inconsistent.

Such flaws and unexpected or undesirable deductions are not properly recognised and implemented in explanation features by the extant reasoners and ontology development environments and therewith do not point to the actual flaw in the object property box. This is primarily because implemented justification and explanation algorithms, such as [6, 2, 7], focus only on logical deductions and errors, and take for granted that class axioms and assertions about instances have to take precedence over what 'ought to be' regarding the object property axioms. Therefore, with the standard reasoning, the object property expressions (inclusion axioms)—i.e., property hierarchy, domain and range axioms, a property's characteristics, and property chains—are assumed to be correct, but instances and classes can move about in the taxonomy. However, the modeller may well be certain where in the taxonomy a particular class belongs, or at least its main category, but not so sure about how to represent its properties. This is a reasonable assumption, given that many ontologies commence with just a bare taxonomy and only gradually add properties, and the advanced OWL 2 DL features for object properties are still relatively new. Therewith it has become an imperative to look at how one can get the modeller to choose the ontologically correct options in the object property box so as to achieve a better quality ontology and, in case of flaws, how to guide the modeller to the root defect from the modeller's viewpoint, and propose corrections. Overall, this requires one to be able to recognise the flaw, to explain it, and to suggest revisions.

We address these issues by introducing two non-standard reasoning services. First, we extend the RBox Compatibility Service for object subproperties from [8] to also handle the object property characteristics, called Sub-Property compatibility Service (*SubProS*), and, second, we define a new ontological reasoning service, Property Chain compatibility Service, (*ProChainS*), that checks whether the chain's properties are compatible. The compatibility services are defined in such a way as to exhaustively check all permutations and therewith pinpoint to the root cause in the object property box, where applicable. If a test of either service fails, proposals are made to revise the identified flaw. As such, *SubProS* and *ProChainS* can be considered *ontological reasoning services*, because the ontology does not necessarily contain logical errors in some of the flaws detected. The solution thus falls in the category of tools focussing on both logic and additional ontology quality criteria, alike the works on anti-patterns [4] and OntoClean [9], by aiming toward ontological correctness in addition to just a satisfiable logical theory. Hence, it is different from other works on explanation and pinpointing mistakes that concern logical consequences only [6, 2, 7], and *SubProS* and *ProChainS* also propose revisions for the flaws.

In the remainder of the paper, we address property subsumption and *SubProS* in Section 2, and property chaining with *ProChainS* in Section 3. We conclude in Section 4.

2 Sub-Properties in OWL

2.1 Preliminaries

Subproperties in OWL have a "basic form" and a "more complex form". The former is denoted in OWL 2 functional syntax as `SubObjectPropertyOf(OPE`$_1$ `OPE`$_2$`)`, which says that object property expression `OPE`$_1$ is a subproperty of `OPE`$_2$, meaning that "if an individual x is connected by OPE$_1$ to an individual y, then x is also connected by OPE$_2$ to y" [1]. The simple version is denoted in Description Logics (DL) as $S \sqsubseteq R$ and a typical use case is properPartOf $\sqsubseteq$ partOf. The more complex form concerns property chains, denoted with `SubObjectPropertyOf(ObjectPropertyChain(OPE`$_1$ `... OPE`$_n$`) OPE)` in OWL 2 and several additional syntactic constraints hold. It is called "complex role inclusions" in DL, and is defined succinctly in [10]. More generally, (sub-)properties are constrained by the Role Inclusion Axioms as defined in [10] for $\mathcal{SROIQ}$, the base language for OWL 2 DL, which also provide the constraints for property chains in OWL 2, and is included below as Definition 1. Informally, case 1 covers transitivity of R, case 2 inverses, case 3 chaining of simple object properties provided the regularity constraint hold (a strict order) so as to maintain decidability, and for case 4 and 5, the property on the right-hand side either occurs first or last in the chain on the left-hand side of the inclusion axiom.

Definition 1 ((Regular) Role Inclusion Axioms ([10])). *Let $\prec$ be a regular order on roles. A* ***role inclusion axiom*** *(RIA for short) is an expression of the form $w \sqsubseteq R$, where w is a finite string of roles not including the universal role U, and $R \neq U$ is a role name. A* ***role hierarchy*** *$\mathcal{R}_h$ is a finite set of RIAs. An interpretation $\mathcal{I}$* ***satisfies*** *a role inclusion axiom $w \sqsubseteq R$, written $\mathcal{I} \models w \sqsubseteq R$, if $w^{\mathcal{I}} \subseteq R^{\mathcal{I}}$. An interpretation is a* ***model*** *of a role hierarchy $\mathcal{R}_h$ if it satisfies all RIAs in $\mathcal{R}_h$, written $\mathcal{I} \models \mathcal{R}_h$. A RIA $w \sqsubseteq R$ is* ***$\prec$-regular*** *if R is a role name, and*

1. *$w = R \circ R$, or*
2. *$w = R^-$, or*
3. *$w = S_1 \circ \ldots \circ S_n$ and $S_i \prec R$, for all $1 \geq i \geq n$, or*
4. *$w = R \circ S_1 \circ \ldots \circ S_n$ and $S_i \prec R$, for all $1 \geq i \geq n$, or*
5. *$w = S_1 \circ \ldots \circ S_n \circ R$ and $S_i \prec R$, for all $1 \geq i \geq n$.*

Finally, a role hierarchy $\mathcal{R}_h$ is ***regular*** *if there exists a regular order $\prec$ such that each RIA in $\mathcal{R}_h$ is $\prec$-regular.*

For reasons of conciseness and readability, we shall use this notation and "$\circ$" for chaining rather than the wordy OWL functional style syntax.

In the remainder of this section, we look into the "basic form" for subproperties, i.e., $S \sqsubseteq R$, and consider property chains (cases 3-5 of Def. 1) in Section 3. To increase readability, we shall use $R \sqsubseteq C_1 \times C_2$ as shortcut for domain and range axioms $\exists R \sqsubseteq C_1$ and $\exists R^- \sqsubseteq C_2$ where C_1 and C_2 are generic classes—`ObjectPropertyDomain(OPE CE)` and `ObjectPropertyRange(OPE CE)` in OWL, respectively. $R \sqsubseteq \top \times \top$ holds when no explicit domain and range axiom has been declared.

2.2 When a property is a subproperty of another

With class subsumption, the subclass is more constrained than its superclass, which can be applied to properties as well: subsumption for OWL object properties (DL roles) holds if the subsumed property is more constrained such that in every model, the set of individual property assertions is a subset of those of its parent property, or: given $S \sqsubseteq R$, then all individuals in the property assertions involving property S must also be related to each other through property R. There are two ways to constrain a property, where either one suffices: by specifying its domain or range, and by declaring the property's characteristics.

Subsumption due to domain and range axioms. Given a subproperty expression $S \sqsubseteq R$, then in order to have the instances of S to be always a subset of the instances of R, S's domain or range, or both, have to be subclasses of the domain and range of R. Although this might be perceived to have an object-oriented and conceptual data modelling flavour, it is widely used (see also Table 1) and declaring domain and range axioms has certain advantages in automated reasoning as as well constraining the admissible models and therewith being more precise in representing the knowledge. Let us first introduce the notion of *user-defined domain and range classes* for OWL ontologies:

Definition 2 (User-defined Domain and Range Classes). *Let R be an OWL object property and $R \sqsubseteq C_1 \times C_2$ its associated domain and range axiom. Then, with the symbol D_R we indicate the* User-defined Domain *of R—i.e., $D_R = C_1$—and with the symbol R_R we indicate the* User-defined Range *of R—i.e., $R_R = C_2$.*

Thus, we need to specify that it ought to be the case that, given an axiom $S \sqsubseteq R$, $D_S \sqsubseteq D_R$ and $R_S \sqsubseteq R_R$ holds, and propose to the ontologist ways to revise the flaw if this is not the case, as it may lead to undesired, or at least 'unexpected', behaviour of the reasoner, being that either the domain class D_S is classified elsewhere in the hierarchy as a subclass of D_R, or, if the classes were declared disjoint, then D_S becomes inconsistent. This was addressed in detail in [8] and solved with the *RBox Compatibility* service, for which the DL $\mathcal{ALCI}$ sufficed to define it. We shall adapt it to OWL 2 DL and extend that service and options to correct it in the next section.

Subsumption due to the property's characteristics. Relational properties—OWL object property characteristics—constrain the way objects relate to each other; e.g., if an ABox contains $\mathsf{connectedTo(a,b)}$, then only if $\mathsf{connectedTo}$ is (correctly) asserted to be symmetric then it will infer $\mathsf{connectedTo(b,a)}$. One can argue for a property hierarchy with respect to the characteristics of the properties; e.g., a property is asymmetric if it is both antisymmetric and irreflexive, hence, if a property is asymmetric, it certainly is also irreflexive but not vv. Again, a subproperty has be more constrained than its parent property and, as with inheritance of properties for classes, the property's characteristic(s) should be inherited along the property hierarchy or overridden with a stronger constraint. With this line of reasoning and $S \sqsubseteq R$, then, e.g., $\mathsf{Asym}(S)$ and $\mathsf{Irr}(R)$ is admissible, but the other way around, $\mathsf{Irr}(S)$ and $\mathsf{Asym}(R)$ is a flaw where either both are, or neither one is, asymmetric or $R \sqsubseteq S$ would be the intended axiom. One can observe this also for the commonly used parthood and proper parthood relations: the former is reflexive, antisymmetric, and transitive, and the latter irreflexive, asymmetric, and transitive. Conversely, direct parthood (dpo) is intransitive, yet tends to be represented in an OWL ontology as a subproperty of parthood (po). However, one cannot represent intransitivity in OWL explicitly (not asserting transitivity means a property is non-transitive, not intransitive).

With respect to OWL 2 DL, the constraints become fairly straight-forward with respect to a property hierarchy, because antisymmetry, acyclicity (which is more constrained than asymmetric), and intransitivity cannot be represented. However, current OWL reasoners do not take into account inheritance of property characteristics even of the same relational property, except transitivity. That is, if, e.g., $\mathsf{Sym}(R)$ then this does not hold automatically for S, which is easy to check with any ontology editor: add $S \sqsubseteq R$, $\mathsf{Sym}(R)$, $C \sqsubseteq \exists R.D$, $E \sqsubseteq \exists S.F$, $r(c_1, d_1)$ and $s(e_1, f_1)$ and observe the inferences. However, in such a setting, S is non-symmetric, which is generally unintended, especially if there is some T where $T \sqsubseteq S \sqsubseteq R$ and $\mathsf{Sym}(T)$ and $\mathsf{Sym}(R)$. Note that adding $\mathsf{Asym}(S)$ is very well possible. This holds for Ref and Irr, too.

2.3 The Sub-Property compatibility Service

We now can define the new reasoning service, *Sub-Property compatibility Service* (*SubProS*) by extending the basic notions from the *RBox compatibility* [8]. Informally, it first checks the 'compatibility' of domain and range axioms with respect to the object property hierarchy and the class hierarchy in the ontology. The RBox compatibility service is already necessary and sufficient for finding domain/range problems, because it exhaustively checks each permutation of domain and range of the parent and child property in the object property hierarchy. After that, *SubProS* checks whether the object property characteristic(s) conform to specification, provided there is such an expression in the ontology.

Definition 3 (Sub-Property compatibility Service (*SubProS*)). *For each pair of object properties, $R, S \in \mathcal{O}$ such that $\mathcal{O} \models S \sqsubseteq R$, and $\mathcal{O}$ an OWL ontology adhering to the syntax and semantics as specified in [1], check whether:*

Test 1. $\mathcal{O} \models D_S \sqsubseteq D_R$ *and* $\mathcal{O} \models R_S \sqsubseteq R_R$;
Test 2. $\mathcal{O} \not\models D_R \sqsubseteq D_S$;
Test 3. $\mathcal{O} \not\models R_R \sqsubseteq R_S$;
Test 4. *If* $\mathcal{O} \models \mathsf{Asym}(R)$ *then* $\mathcal{O} \models \mathsf{Asym}(S)$;
Test 5. *If* $\mathcal{O} \models \mathsf{Sym}(R)$ *then* $\mathcal{O} \models \mathsf{Sym}(S)$ *or* $\mathcal{O} \models \mathsf{Asym}(S)$;
Test 6. *If* $\mathcal{O} \models \mathsf{Trans}(R)$ *then* $\mathcal{O} \models \mathsf{Trans}(S)$;
Test 7. *If* $\mathcal{O} \models \mathsf{Ref}(R)$ *then* $\mathcal{O} \models \mathsf{Ref}(S)$ *or* $\mathcal{O} \models \mathsf{Irr}(S)$;
Test 8. *If* $\mathcal{O} \models \mathsf{Irr}(R)$ *then* $\mathcal{O} \models \mathsf{Irr}(S)$ *or* $\mathcal{O} \models \mathsf{Asym}(S)$;

Test 9. *If* $\mathcal{O} \models \mathsf{Asym}(R)$ *then* $\mathcal{O} \not\models \mathsf{Sym}(S)$*;*
Test 10. *If* $\mathcal{O} \models \mathsf{Irr}(R)$ *then* $\mathcal{O} \not\models \mathsf{Ref}(S)$*;*
Test 11. *If* $\mathcal{O} \models \mathsf{Trans}(R)$ *then* $\mathcal{O} \not\models \mathsf{Irr}(R)$*,* $\mathcal{O} \not\models \mathsf{Asym}(R)$*,* $\mathcal{O} \not\models \mathsf{Irr}(S)$*, and* $\mathcal{O} \not\models \mathsf{Asym}(S)$*;*

An OWL object property hierarchy is said to be compatible iff
- `Test 1` *and (*`2` *or* `3`*) hold for all pairs of property-subproperty in* $\mathcal{O}$*, and*
- `Tests 4-11` *hold for all pairs of property-subproperty in* $\mathcal{O}$*.*

An OWL ontology $\mathcal{O}$ that does not adhere to *SubProS* is considered to be *ontologically flawed* and is in need of repair. To arrive at the repair, we can avail in part of the extant OWL reasoners. The class subsumption checking for `Tests 1-3` can be done with any of the OWL reasoners, where the result is processed to check the conditions in the tests. `Test 9` and `Test 10` are already done by OWL 2 DL reasoners, but it now only returns a class inconsistency when S is used in a class expression, not regarding the property characteristics per sé; hence, the explanation and suggestions for revision has to be amended: with respect to the intended meaning, not the class expression is at fault, but there is a flaw in the property hierarchy. `Test 11` is included merely for purpose of exhaustiveness, as it is already prohibited by the syntax restrictions of OWL 2 and already has a corresponding warning (irreflexivity and asymmetry require simple object properties, but a transitive property is not simple), but it may become relevant for any future OWL version or modification of *SubProS* for another ontology language. By the same token, one could argue to remove `Test 6`, for it is already computed, but it makes it easier for any extensions of the service or its adoption for another ontology language.

The following sequence specifies what has to be done if any of the applicable tests fails. For `Tests 1-3`, we can reuse the basic 2-step idea from [8] and adapt it to the current setting, and we propose new corrections for `Tests 4-11`. Observe that "raising a warning" denotes that it is not a logical error but an ontological one, "forcing" a revision indicates there is a logical error that must be fixed in order to have a consistent ontology with satisfiable classes, and "propose" indicates suggestions how the flaw can be best revised.

A. If `Test 1` fails, raise a warning "domain and range restrictions of either R or S are in conflict with the property hierarchy", and propose to
   - ⋆ Change the object property hierarchy, i.e., either remove $S \sqsubseteq R$ and add $R \sqsubseteq S$ or add $S \equiv R$ to $\mathcal{O}$, or
   - ⋆ Change domain and range restrictions of R and/or S, or
   - ⋆ If the test on the domains fails, then propose a new axiom $R \sqsubseteq D'_R \times R_R$, where $D'_R \equiv D_R \sqcap D_S$ (and similarly when `Test 1` fails on the range).

B. If `Test 2` and `Test 3` fail, raise a warning "R cannot be a proper subproperty of S, but they can be equivalent", and propose:
   - ⋆ Accept the possible equivalence and, optionally, add $S \equiv R$ to $\mathcal{O}$, or
   - ⋆ Change domain and range restrictions of R and/or S.

C. Run *SubProS* again if any changes have been made in steps A or B, and record changes in the hierarchy (to be used in step I).

D. If $\mathsf{Asym}(R)$ is asserted in $\mathcal{O}$ and `Test 4` fails to detect $\mathsf{Asym}(S)$, raise a warning "R is asymmetric, but its subproperty S is not", proposing to remedy this with either:
   ⋆ Add $\mathsf{Asym}(S)$ to obtain expected inferences;
   ⋆ Remove $\mathsf{Asym}(R)$;
   ⋆ Change the positions of R and/or S in the object property hierarchy;
   and similarly when `Test 6` fails,
E. If $\mathsf{Sym}(R)$ is asserted and `Test 5` fails to detect either $\mathsf{Sym}(S)$ or $\mathsf{Asym}(S)$, raise a warning "R is symmetric, but its subproperty S is not, nor is it asymmetric", proposing to remedy this with either:
   ⋆ Add $\mathsf{Sym}(S)$ or $\mathsf{Asym}(S)$ to obtain expected inferences;
   ⋆ Remove $\mathsf{Sym}(R)$;
   ⋆ Change the positions of R and/or S in the object property hierarchy;
   and similarly when `Test 7` fails,
F. If $\mathsf{Irr}(R)$ and `Test 8` fails to detect either $\mathsf{Irr}(S)$ or $\mathsf{Asym}(S)$, raise a warning "R is irreflexive, hence S should be either $\mathsf{Irr}(S)$ or $\mathsf{Asym}(S)$, and propose:
   ⋆ Add $\mathsf{Asym}(S)$ or $\mathsf{Irr}(S)$ to obtain expected inferences;
   ⋆ Remove $\mathsf{Irr}(R)$;
   ⋆ Change the positions of R and/or S in the object property hierarchy;
G. If `Test 9` fails, report "R is asymmetric so its subproperty, S, cannot be symmetric" and force the modeller to change it by either
   ⋆ Remove $\mathsf{Asym}(R)$, or
   ⋆ Remove $\mathsf{Sym}(S)$.
   ⋆ Change the positions of R and/or S in the object property hierarchy;
   and similarly if `Test 10` fails, but then irreflexive and reflexive, respectively.
H. If `Test 11` fails, report "R (and by `Test 6`, S, too) is declared transitive, hence, not a simple object property, hence it is not permitted to participate in an irreflexive or asymmetric object property expression" and force the modeller to change it by either:
   ⋆ Remove $\mathsf{Trans}(R)$, or
   ⋆ Remove $\mathsf{Irr}(R)$, $\mathsf{Asym}(R)$, $\mathsf{Irr}(S)$, and $\mathsf{Asym}(S)$;
I. Run *SubProS* again if any changes have been made in steps D-H, and check any changes in the property hierarchy made against those recorded in step C. If a change from steps E or F reverts a recorded change, then report "unresolvable conflict on subproperty axiom. You *must* change at least one axiom to exit an otherwise infinite loop of swapping two expressions".

The reason for running *SubProS* again after `Test 1-3` and not only at the end is that those changes, if any, will affect the outcome of `Tests 4-11`, and in Step I to both push through the changes and prevent an infinite loop. *SubProS* was evaluated with several ontologies, one of which illustrated now.

Evaluation 1 (BioTop's inconsistent 'has process role') The TONES Ontology repository [`http://owl.cs.manchester.ac.uk/repository/`] contains 219 ontologies, of which we selected 10 ontologies semi-randomly based on having listed in the metrics to have a substantial amount of object properties, and being a real ontology (more precisely: no toy or tutorial ontology, not converted from OBO, nor an OWLized thesaurus). Some relevant data is shown in Table 1. Interesting to observe is the plethora of object properties that are mostly in an hierarchy and predominantly based on changes in the domain or range. Due to space limitations, we shall analyse here only BioTop's inconsistent object property.

The 'has process role' in BioTop [11] version d.d. June 17, 2010 (last update) is inconsistent, yet there is no explanation for it computed by currently implemented explanation algorithms. We have in the ontology, among other things: 'has process role'$\sqsubseteq$'temporally related to', 'has process role'$\sqsubseteq$'processual entity'$\times$role, 'temporally related to' $\sqsubseteq$ 'processual entity'$\sqcup$quality $\times$ 'processual entity'$\sqcup$quality, role $\sqsubseteq \neg$quality, role $\sqsubseteq \neg$'processual entity', Sym('temporally related to').
Let us use *SubProS* to isolate the error and propose a revision.

- `Test 1` fails, because $\mathcal{O} \not\models R_{\texttt{hasprocessrole}} \sqsubseteq R_{\texttt{'temporallyrelatedto'}}$, as the ranges are disjoint;
- `Test 2` and `3` pass.
- `Test 4` is not applicable.
- `Test 5` fails, because $\mathcal{O}$ does not contain Sym('has process role').
- `Test 6-11` are not applicable.

To remedy the issue with `Test 1`, we have three options (see item `A`, above). Thus far, 'has process role' has not been used in class expressions, and from the intention described in the annotation, it suits as subproperty of 'temporally related to', therefore, one opts for choice 2, being to change the range of 'temporally related to' into 'processual entity' $\sqcup$ quality $\sqcup$ role. (The same holds for the inverse 'process role of'.). Second, with `Test 5` failing, we have three options to revise the flaw (see item `E`, above). We already decided not to change the position of the property in the hierarchy, so we either add Sym('has process role') or Asym('has process role') or remove Sym('temporally related to'). Ontologically, Sym('temporally related to') is certainly true and Sym('has process role') certainly false, hence Asym('has process role') is the appropriate choice. After making these changes, we run *SubProS* again, and no issues emerge. $\diamondsuit$

3 Property Chaining in OWL

Property chaining is well-known in DL research as role composition [12–14], but in OWL its usage is more restricted and, except for a few elaborate cases (the family relations example [1] and metamodelling rewritings [15]), is typically used with only two object properties being chained (e.g., [5, 16–18]). The issues mentioned and solution proposed here applies to any OWL 2 DL admissible chain. Let us introduce an example of an undesirable deduction first.

Example 1. (DMOP ontology) The Data Mining and OPtimisation (DMOP) ontology of the e-LICO project (`http://www.e-lico.eu`) v5.2 (Sept. 2011) includes 11 property chains, including hasMainTable $\circ$ hasFeature $\sqsubseteq$ hasFeature, where the properties have domain and range axioms hasMainTable $\sqsubseteq$ DataSet$\times$ DataTable and hasFeature $\sqsubseteq$ DataTable $\times$ Feature, which is depicted in Fig. 1.

Table 1. Selection of some TONES Repository ontologies, retrieved on 12-3-2012 (see *Evaluation 1* for details); OP = object property; char. = object property characteristic.

Ontology	**No. of OPs**	**No. of SubOPs axioms**	**No. more constrained by D or R**	**No. more constrained by char.**	**Comments** (partial)
DOLCE-lite	70	46	13	3	transitivity added
SAO 1.2	36	25	21	5	2 x transitivity; `Test 6` fails on has Vesicle Component
airsystem	111	56	43	2	imports DUL. *ProChainS* fails
process (SWEET)	102	10	7	0	
family-tree	52	25	14	2	fails `Test 6` of *SubProS*
propreo	32	20	17	2	beyond OWL 2 DL (non-simple prop. in max card.)
heart	29	18	9	0	many inconsistencies
mygrid-unclassified	69	39	0	3	1 x transitive added
building Architecture	28	24	0	0	imports rcc, fails `Test 5` of *SubProS* (omission Asym on properPartOf)
biotop	89	84	45	9	with transitivity; 'has process role' is inconsistent, see *Evaluation 1*

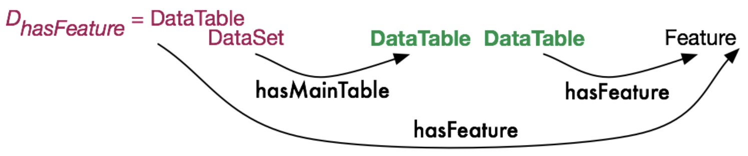


Fig. 1. Graphical depiction of the property chain hasMainTable ∘ hasFeature ⊑ hasFeature with the domain and range axioms of the two object properties.

Thus, while the range of hasMainTable and domain of hasFeature match neatly—both are DataTable—this is not the case for their respective domains, because $D_{\text{hasMainTable}}$ = DataSet and $D_{\text{hasFeature}}$ = DataTable. Together with the represented knowledge that DataSet is a not-disjoint sister-class of DataTable, the combination of the domain and range axioms and the property chain causes the deduction that DataSet ⊑ DataTable, which is wrong with respect to the subject domain semantics. Put differently: the real flaw is either in the domain and range axioms of either one of the two object properties, or there is a flaw in the chain axiom. We will revisit this issue in Evaluation 4. ♢

To foster development of good quality ontologies, the ontologist should at least be informed about probable modelling flaws and be guided in the right direction to revise axioms in some way. This requires introduction of new constraints on the use of property chains in order to guarantee correct and meaningful reasoning with respect to the subject domain, and explanation of the derivations. We shall address the issues with a non-standard reasoning service, called the *Property Chain Compatibility Service* (*ProChainS*).

3.1 The Property Chain Compatibility Service

Given Definition 1 on role inclusion axioms, we need to consider cases 3, 4, and 5 and, without loss of generality, for each OWL object property, if a domain and range is declared, exactly one domain and range axiom is provided. Recall the notation as in Definition 1 and 2: e.g., for Case 3, we may have a property chain $S_1 \circ S_2 \circ S_3 \sqsubseteq R$ where each property has corresponding domain and range D_{S1}, R_{S1}, D_{S2}, R_{S2}, D_{S3}, R_{S3}, D_R, and R_R. The three cases with the constraints that must hold are described formally in Definition 4, which concerns principally how the domain and range of the object properties used in the property chain expressions should relate with respect to their position in the class hierarchy.

Definition 4 (Property Chain Compatibility Service (*ProChainS*)). *For each set of object properties, $R, S_1, \ldots, S_n \in \mathcal{R}$, $\mathcal{R}$ the set of OWL object properties (V_{OP} in [1]) in OWL ontology $\mathcal{O}$, and $S_i \prec R$ with $1 \leq i \leq n$, $\mathcal{O}$ adheres to the constraints of Definition 1 (and, more generally, the OWL 2 specification [1]), and user-defined domain and range axioms as defined in Definition 2, for each of the property chain expression, select either one of the three cases:*

Case S. *Property chain pattern as $S_1 \circ S_2 \circ \ldots \circ S_n \sqsubseteq R$. Test whether:*
  Test S-a. $\mathcal{O} \models R_{S1} \sqsubseteq D_{S2}, \ldots, R_{Sn-1} \sqsubseteq D_{Sn}$;
  Test S-b. $\mathcal{O} \models D_{S1} \sqsubseteq D_R$;
  Test S-c. $\mathcal{O} \models R_{Sn} \sqsubseteq R_R$;

Case RS. *Property chain pattern as $R \circ S_1 \circ \ldots \circ S_n \sqsubseteq R$. Test whether:*
  Test RS-a. $\mathcal{O} \models R_{S1} \sqsubseteq D_{S2}, \ldots, R_{Sn-1} \sqsubseteq D_{Sn}$;
  Test RS-b. $\mathcal{O} \models R_R \sqsubseteq D_{S1}$;
  Test RS-c. $\mathcal{O} \models R_{Sn} \sqsubseteq R_R$;

Case SR. *Property chain pattern as $S_1 \circ \ldots \circ S_n \circ R \sqsubseteq R$. Test whether:*
  Test SR-a. $\mathcal{O} \models R_{S1} \sqsubseteq D_{S2}, \ldots, R_{Sn-1} \sqsubseteq D_{Sn}$;
  Test SR-b. $\mathcal{O} \models D_{S1} \sqsubseteq D_R$;
  Test SR-c. $\mathcal{O} \models R_{Sn} \sqsubseteq D_R$;

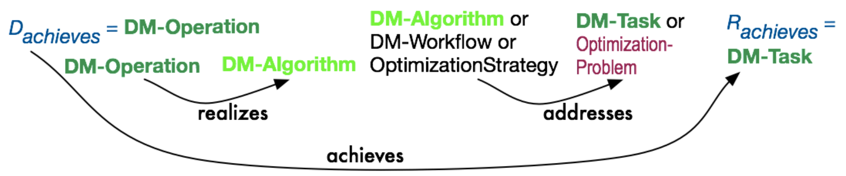
An OWL property chain expression is said to be compatible iff the OWL 2 syntactic constraints hold and either `Case S`*, or* `Case RS`*, or* `Case SR` *holds.*

ProChainS is tested against several domain ontologies in the following two evaluations. To simplify the explanations, let us assume that each ontology $\mathcal{O}$ contains an OWLized DOLCE taxonomy (relevant are PD $\sqsubseteq$ PT, ED $\sqsubseteq$ PT, NPED $\sqsubseteq$ ED, ED $\sqsubseteq \neg$PD, NPED $\sqsubseteq$ PED, POB $\sqsubseteq$ PED, and NPED $\sqsubseteq \neg$POB), subject domain classes (e.g. DrugTreatment) involved in class expressions, DOLCE's hasPart $\sqsubseteq$ PT $\times$ PT and hasParticipant $\sqsubseteq$ PD $\times$ ED, a part-of between perdurants (involvedIn $\sqsubseteq$ PD $\times$ PD), and a structuralPart $\sqsubseteq$ POB $\times$ POB.

Evaluation 2 (DMOP chains, Case S) The aforementioned DMOP ontology v5.2 contains a property chain realizes ∘ addresses ⊑ achieves, and several domain and range axioms for the properties, which is depicted in Fig. 2; the classes are all subclasses of DOLCE's NPED. Going through the three applicable tests of *ProChainS*, we obtain:

– `Test S-a` passes, because $R_{\texttt{realizes}} \sqsubseteq D_{\texttt{addresses}}$;
– `Test S-b`, "$D_{\texttt{realizes}} \sqsubseteq D_{\texttt{achieves}}$?", holds: they are both DM-Operation;
– `Test S-c`, "$R_{\texttt{addresses}} \sqsubseteq R_{\texttt{achieves}}$?", fails because $R_{\texttt{addresses}}$ is the union of DM-Task and OptimizationProblem.

Thus, the chain can be instantiated, but if the range of addresses in a class expression is a subclass of OptimizationProblem, then all its instances will be classified as being a member of DM-Task as well, given that the two classes are not declared disjoint. If the two classes would have been declared disjoint, then the ontology would have become inconsistent, with as root problem a "bad individual" member of DM-Operation, instead of pointing to the issue with $R_{\texttt{addresses}}$ and $R_{\texttt{achieves}}$. In fact, even the current classification is undesirable: tasks and problems are clearly distinct entities. In this case, the lead ontology developer chose to revise the domain and range restrictions of the addresses property to have the chain functioning as intended (included in v5.3, d.d. 10-1-2012)[2]. ♢



$D_{achieves}$ = DM-Operation
DM-Operation
DM-Algorithm
realizes
DM-Algorithm or DM-Workflow or OptimizationStrategy
DM-Task or Optimization-Problem
addresses
$R_{achieves}$ = DM-Task
achieves

Fig. 2. The realizes ∘ addresses ⊑ achieves chain in DMOP v5.2, with the domain and range axioms of the participating properties; the matching chain is indicated in green, the problematic one in Arial narrow maroon font.

Evaluation 3 (Pharmacogenomics chains, Case SR) Reconsider the pharmacogenomics ontology with the chain hasPart ∘ hasParticipant ⊑ hasParticipant and its use with drugs and treatments [5], and aforementioned axioms. With *ProChainS*:

– `Test SR-a` is trivially satisfied ($i = 1$);
– `Test SR-b` "$D_{\texttt{hasPart}} \sqsubseteq D_{\texttt{hasParticipant}}$?" does not hold because PD ⊑ PT;
– `Test SR-c` "$R_{\texttt{hasPart}} \sqsubseteq D_{\texttt{hasParticipant}}$?" does not hold because PD ⊑ PT.

If $\mathcal{O} \models$ DrugTreatment ⊑ PT and $\mathcal{O} \models$ DrugGeneInteraction ⊑ PT—not included in [5], however—then the OWL reasoners will infer they are subclasses of PD.

[2] Whether this is the best option in an absolute sense is a separate issue; e.g., one can reify the relations and have whatever is achieved be subsumed by DOLCE's Achievement or whatever is realised be subsumed by BFO's Realizable and add corresponding class expressions for the participating classes.

If we add $\mathsf{DrugTreatment} \sqsubseteq \mathsf{ED}$ and $\mathsf{DrugGeneInteraction} \sqsubseteq \mathsf{ED}$ to $\mathcal{O}$, then it deduces that the two classes are inconsistent because $\mathsf{ED} \sqsubseteq \neg\mathsf{PD}$. Dumontier's hasPart thus holds only if DrugTreatment and DrugGeneInteraction are subclasses of PD (perdurants or 'processes'), hence, we can refine the property chain into $\mathsf{involvedIn}^{-} \circ \mathsf{hasParticipant} \sqsubseteq \mathsf{hasParticipant}$. Assessing the tests for *Case SR* again, then `Test SR-b` and `Test SR-c` do hold, because $D_{\texttt{involvedIn}^{-}} = R_{\texttt{involvedIn}^{-}} = D_{\texttt{hasParticipant}} = \mathsf{PD}$. Thus, we now have a property chain that is *guaranteed* not to lead to an inconsistency when the object properties are used in OWL class expressions. ♢

3.2 Managing Consequences of Property Chains

As Evaluation 3 shows, the ontology may not necessarily be inconsistent when viewed purely from a logic perspective, and, in fact, classify one or more of the participating classes elsewhere in the taxonomy with respect to where it was added originally (be this ontologically correct or not). Put differently, one cannot enforce *ProChainS*'s outcomes on the user. Be they undesired inferences or inconsistencies in the class hierarchy, it is important to have an explanation that those consequences are due to the property chain.

Now that we know what and how to check whether a declared property chain is logically and ontologically correct, it is also possible to devise support for identifying modelling defects, communicating this to the user, and suggest options to correct it in a similar way as for *SubProS*. Given that it can be considered an ontological reasoning service of one aspect of the ontology only, a less comprehensive approach can be taken compared to the formal foundations of computing explanations or root justifications [6, 2], as in this case we do not have to find the root anymore and, in fact, can make use of certain explanation features that are implemented already. We propose the following ontology updates, in case any of the tests fails:

A. If `Test S-a`, `Test RS-a`, or `Test SR-a` fails, check for any violating pair R_{Si}, D_{Si+1} whether:
   (i) $\mathcal{O} \models R_{Si} \sqsubseteq \neg D_{Si+1}$, then raise a warning "Incompatible domain and range of R_{Si}, D_{Si+1} in the property chain expression. This is *certain* to lead to an inconsistent class when the properties are used in class axioms, and an inconsistent ontology when used in assertions about instances", and propose the following minimal corrections:
      ⋆ Change the range of S_i such that $\mathcal{O} \models R_{Si} \sqsubseteq D_{Si+1}$, or
      ⋆ Change the domain of S_{i+1} such that $\mathcal{O} \models R_{Si} \sqsubseteq D_{Si+1}$;
      ⋆ Change the property chain such that a compatible property participates;
   (ii) $\mathcal{O} \models D_{Si+1} \sqsubseteq R_{Si}$, then raise a warning "Incompatible domain and range of R_{Si}, D_{Si+1} in the property chain expression. This *either* results in an inconsistent class when the properties are used in class axioms and an inconsistent ontology when used in assertions about instances, *or* results in a classification of D_{Si+1} elsewhere in the class hierarchy", and propose the following minimal corrections:

  - ⋆ Change the range of S_i such that $\mathcal{O} \models R_{Si} \sqsubseteq D_{Si+1}$, or
  - ⋆ Change the domain of S_{i+1} such that $\mathcal{O} \models R_{Si} \sqsubseteq D_{Si+1}$;
  - ⋆ Change the property chain such that a compatible property participates;
  - ⋆ Let the reasoner classify D_R as a subclass of D_{S1} and accept this inference, provided $\mathcal{O} \not\models D_R \sqsubseteq \bot$;

B. If `Test S-b` fails, then raise a warning "Incompatible domain and range of D_{S1}, D_R in the property chain expression, which will induce a classification of D_R elsewhere in the taxonomy or an inconsistency" and propose the following options:
  - ⋆ Change the domain of R or S_1 such that $\mathcal{O} \models D_{S1} \sqsubseteq D_R$, or
  - ⋆ Let the reasoner classify D_{Si+1} as a subclass of R_{Si} and accept this inference, provided $\mathcal{O} \not\models D_{Si+1} \sqsubseteq \bot$;

  and similarly for the respective ranges of R and S_n in `Test S-c`.

C. If `Test RS-b` fails, then raise a warning "Incompatible domain and range of D_{S1}, R_R in the left-hand-side of the property chain expression, which will induce a classification of R_R elsewhere in the taxonomy or in inconsistency" and propose:
  - ⋆ Change the domain of S_1 or range of R such that $\mathcal{O} \models D_{S1} \sqsubseteq R_R$, or
  - ⋆ Let the reasoner classify R_R as a subclass of D_{S1} and accept this inference, provided $\mathcal{O} \not\models R_R \sqsubseteq \bot$;

  and similarly for the respective ranges of R and S_n in `Test RS-c`.

D. If `Test SR-b` fails then raise a warning "Incompatible domain and range of D_{S1}, D_R in the property chain expression, which will induce a reclassification or inconsistency of D_{S1}" and propose the following options:
  - ⋆ Change the domain of S_1 or R such that $\mathcal{O} \models D_{S1} \sqsubseteq D_R$, or
  - ⋆ Let the reasoner classify D_{S1} as a subclass of R_R and accept this inference, provided $\mathcal{O} \not\models D_{S1} \sqsubseteq \bot$;

  and similarly for the range of S_n (compared to the range of R) in `Test SR-c`.

E. Run *ProChainS* again if any changes have been made in steps A-D.

ProChainS and the management of its consequences is evaluated with the DMOP ontology.

Evaluation 4 (*Assessing DMOP chains*) Recollect the property chain problem described in Example 1 with DMOP v5.2. DMOP uses the $\mathcal{SROIQ}$ features, has some 573 classes, 1021 subclass axioms, 113 object properties, and 11 property chains. Eight chains raise a warning with *ProChainS*, of which 3 cause a classification of classes elsewhere in the taxonomy due to the chain expressions. Among others, there is $\mathsf{hasMainTable} \circ \mathsf{hasFeature} \sqsubseteq \mathsf{hasFeature}$ of Example 1, which is an instance of *Case SR*.

– `Test SR-a` passes trivially, for $i = 1$.
– `Test SR-b` fails: $D_{\mathsf{hasMainTable}} = \mathsf{DataSet}$ and $D_{\mathsf{hasFeature}} = \mathsf{DataTable}$, but $\mathsf{DataSet}$ is a not-disjoint sister-class of $\mathsf{DataTable}$, so $\mathcal{O} \not\models D_{\mathsf{hasMainTable}} \sqsubseteq D_{\mathsf{hasFeature}}$, therefore `Test SR-b` fails and $\mathsf{DataSet} \sqsubseteq \mathsf{DataTable}$ is deduced, which is deemed undesirable. Step D's suggestions to revise the ontology are either to change the domain or to accept the new classification; the DMOP

domain experts chose the first option for revision and changed the domain of hasFeature into DataSet $\sqcup$ DataTable, which is included in DMOP v5.3.
– `Test SR-c` passes, as both are DataTable.

No inconsistencies or unexpected classifications were detected with the other five chains, mainly thanks to the absence of disjointness axioms. For instance, the *Case S* described in Evaluation 2 with realizes $\circ$ addresses $\sqsubseteq$ achieves: `Test S-a` and `S-b` pass, but `S-c` does not, because $\mathcal{O} \not\models R_{\text{DM-Task}\sqcup\text{OptimizationProblem}} \sqsubseteq R_{\text{DM-Task}}$. A subclass of OptimizationProblem together with relevant property expressions declared for the chain results in an undesirable deduction that it is a subclass of DM-Task; hence, step B's first option (for `S-c`) to revise the ontology was the chosen option, i.e., removing OptimizationProblem from the range axiom of addresses (as well as removing OptimizationStrategy from the domain axiom), which is included as such in DMOP v5.3. ♢

Thus, *SubProS* and *ProChainS* together cover all types of modelling flaws with their root causes and options to revise them in OWL ontologies with respect to property hierarchies, domain and range axioms to type the property, a property's characteristics, and property chains.

4 Conclusions

We have identified exhaustively the type of flaws that can occur in the object property box regarding simple property subsumption and property chaining and proposed two compatibility services, *SubProS* and *ProChainS*, that both check for meaningful object property hierarchies and property chaining. Thanks to being able to identify the root cause, proposals for how to revise the ontology were made, including the options to change the object property expressions or the class hierarchy, and how, or accepting the deductions. This was evaluated with several ontologies were flaws could be detected and were solved, therewith improving the ontology's quality.

We are currently looking into an efficient algorithm to implement *SubProS* and *ProChainS* and a user-friendly interface to help revising flaws.

Acknowledgements The author wishes to thank Melanie Hilario for her feedback on the subject domain and object properties in the DMOP ontology.

References

1. Motik, B., Patel-Schneider, P.F., Parsia, B.: OWL 2 web ontology language structural specification and functional-style syntax. W3c recommendation, W3C (27 Oct. 2009) http://www.w3.org/TR/owl2-syntax/.
2. Parsia, B., Sirin, E., Kalyanpur, A.: Debugging OWL ontologies. In: Proceedings of the World Wide Web Conference (WWW 2005). (2005) May 10-14, 2005, Chiba, Japan.

3. Rector, A., Drummond, N., Horridg, e.M., Rogers, L., Knublauch, H., Stevens, R., Wang, H., Wroe, Csallner, C.: OWL pizzas: Practical experience of teaching OWL-DL: Common errors & common patterns. In: Proceedings of the 14th International Conference Knowledge Acquisition, Modeling and Management (EKAW'02). Volume 3257 of LNCS., Springer (2004) 63–81 Whittlebury Hall, UK.
4. Roussey, C., Corcho, O., Vilches-Blázquez, L.: A catalogue of OWL ontology antipatterns. In: Proc. of K-CAP'09. (2009) 205–206
5. Dumontier, M., Villanueva-Rosales, N.: Modeling life science knowledge with OWL 1.1. In: Fourth International Workshop OWL: Experiences and Directions 2008 (OWLED 2008 DC). (2008) Washington, DC (metro), 1-2 April 2008.
6. Horridge, M., Parsia, B., Sattler, U.: Laconic and precise justifications in OWL. In: Proc. of the 7th International Semantic Web Conference (ISWC 2008). Volume 5318 of LNCS., Springer (2008)
7. Kalyanpur, A., Parsia, B., Sirin, E., Grau, B.: Repairing unsatisfiable concepts in OWL ontologies. In: Proceedings of ESWC'06. (2006)
8. Keet, C.M., Artale, A.: Representing and reasoning over a taxonomy of part-whole relations. Applied Ontology **3**(1-2) (2008) 91–110
9. Guarino, N., Welty, C.: An overview of OntoClean. In Staab, S., Studer, R., eds.: Handbook on ontologies. Springer Verlag (2004) 151–159
10. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible $\mathcal{SROIQ}$. Proceedings of KR-2006 (2006) 452–457
11. Beisswanger, E., Schulz, S., Stenzhorn, H., Hahn, U.: BioTop: An upper domain ontology for the life sciences - a description of its current structure, contents, and interfaces to OBO ontologies. Applied Ontology **3**(4) (2008) 205–212
12. Massacci, F.: Decision procedures for expressive description logics with intersection, composition, converse of roles and role identity. In: Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI'2001). (2001) 193–198
13. Schmidt-Schauss, M.: Subsumption in KL-ONE is undecidable. In: Proceedings of 1st Conference on Knowledge Representation and Reasoning (KR'89). (1989) 421–431
14. Wessel, M.: Obstacles on the way to qualitative spatial reasoning with description logics: some undecidability results. In Goble, C.A., McGuinness, D.L., Möller, R., Patel-Schneider, P.F., eds.: Proceedings of the International Workshop in Description Logics (DL'01). Volume 49 of CEUR WS. (2001) Stanford, CA, USA, August 1-3, 2001.
15. Glimm, B., Rudolph, S., Völker, J.: Integrated metamodeling and diagnosis in OWL 2. In Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B., eds.: Proceedings of the 9th International Semantic Web Conference. Volume 6496 of LNCS., Springer (November 2010) 257–272
16. Boran, A., Bedini, I., Matheus, C.J., Patel-Schneider, P.F., Keeney, J.: Choosing between axioms, rules and queries: Experiments in semantic integration techniques. In: Eigth International Workshop OWL: Experiences and Directions (OWLED'11). (2011) San Francisco, California, USA, June 5-6 2011.
17. Koutsomitropoulos, D.A., Solomou, G.D., Papatheodorou, T.S.: Metadata and semantics in digital object collections: A case-study on CIDOC-CRM and Dublin Core and a prototype implementation. Journal of Digital Information **10**(6) (2009)
18. Hilario, M., Nguyen, P., Do, H., Woznica, A., Kalous, A.: Ontology-based meta-mining of knowledge discovery workflows. In Jankowski, N., Duch, W., Grabczewski, K., eds.: Meta-learning in Computational Intelligence. Springer (2011) 273–315