Noname manuscript No.
(will be inserted by the editor)

Preventing, Detecting, and Revising Flaws in Object Property Expressions

C. Maria Keet

Received: date / Accepted: date

Abstract The OWL 2 DL ontology language is very expressive and has many features for declaring complex object property expressions. Standard reasoning services for OWL ontologies take these expressions as 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 in the class hierarchy may actually be due to a mistake in the 'object property box', not the class axioms. We analyse the principles of subsumption in object property hierarchies, and use it to identify the types of flaws that can occur in object property expressions. We propose the compatibility services *SubProS* and *ProChainS* that check for meaningful property hierarchies and property chaining and propose how to revise a flaw. These insights can also be used to prevent flaws and to choose the best option, which we demonstrate with the chain pattern for upward and downward distributivity over parthood relations. *SubProS* and *ProChainS* were evaluated with several ontologies, which demonstrates that such flaws do exist, that they can be isolated effectively, and useful suggestions for revisions can be proposed.

Keywords OWL · Role Inclusion Axioms · Property Chains · Property Hierarchy · Ontology Quality

CR Subject Classification Ontology engineering

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
Tel.: +27-31-2601035
Fax: +27-31-2607001
E-mail: keet@ukzn.ac.za

1 Introduction

There are multiple ongoing ontology development efforts and an increasing amount of projects that require ontology-driven information systems in various subject domains, such as the life sciences, medicine, e-learning, and the enterprise. New ontologies are being developed typically with the most recent W3C-standardised ontology language, OWL 2 [23], which is based on the Description Logics (DL) language $\mathcal{SROIQ}$ [13] that, thanks to domain experts and 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 some syntactic constraints on their usage, but still a lot is possible to represent knowledge precisely. This 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 [24,25,27,28]. We briefly illustrate four different ways how and where mistakes in object property expressions can occur.

(i) *Domain and range flaws in basic hierarchies*; e.g. (simplified), hasParent $\sqsubseteq$ hasMother instead of hasMother $\sqsubseteq$ 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 in basic hierarchies*: e.g. (simplified), connectedTo $\sqsubseteq$ LocatedIn, with Asym(LocatedIn) so that Sym(connectedTo) cannot be asserted anymore although connectedTo is symmetric, or hasGrandFather $\sqsubseteq$ hasAncestor and Trans(hasAncestor) in the `family-tree` ontology[1]

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

but intransitivity of hasGrandFather cannot be asserted.

(iii) *Property chain issues in complex hierarchies*; e.g., hasPart ∘ hasParticipant ⊑ 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.

(iv) *Domain modelling and choosing the right representation from alternatives*; consider the chains haspart ∘ contains ⊑ haspart and contains ∘ haspart ⊑ contains from [4], depicted in Figure 1, with some instances in Figures 1-C and D. Consider option B with example D: it certainly is *not* the case—i.e., an undesirable deduction due to the property chain—that Mary haspart Legominifigure1 (it is not a part like Mary's mouth is part of Mary). Consider option A with C: Mary's mouth contains Lego minifigure1's leg, which is a correct deduction with respect to the subject domain thanks to the property chain. Problematic from a practical viewpoint are 'hidden' chains, such as in the medical terminology SNOMED-CT[2]: congenital absence of one tooth (ID 109442002) implies a congenital absence of mouth (ID 91946007), and likewise for amputations and parthood (toes and therefore feet and so on, ID 95858001), whereas other properties do distribute upwards; e.g., partOf ∘ hasInjury ⊑ hasInjury lets one infer that an injury of the toe is an injury of the foot.

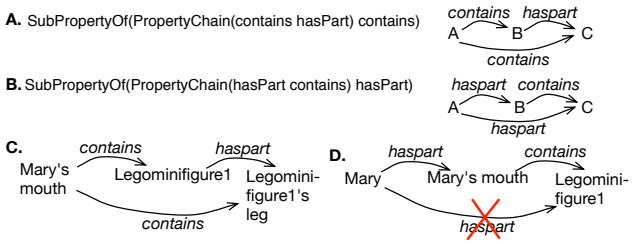

Fig. 1 Example of a disallowed combination of property chains in OWL 2 DL, i.e., either A or B is permitted in OWL 2, but not both together (Source: [4]). Option A is preferred because B is ontologically incorrect, which is illustrated with Mary chewing on a Legominifigure in C and D.

Such flaws and undesirable deductions are not properly recognised by automated reasoners and implemented in explanation features by ontology development environments, as they do not point to the actual flaw in the object property box; this is illustrated in Example 1.

Example 1 (Current automated deductions and explanations) Ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$ contain several object

[2] http://www.ihtsdo.org/our-standards/snomed-ct/

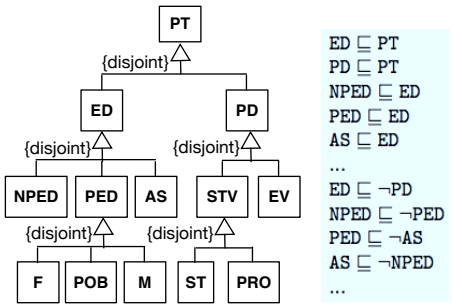

Fig. 2 A section of the DOLCE taxonomy in UML notation, and a selection in DL notation.

property and class expressions, including the formal counterpart of the class hierarchy as depicted in Figure 2 and some subject domain classes, summarised in Table 1. S's domain and range are PED (i.e., S ⊑ PED× PED) and R ⊑ ED × ED is correct with respect to the notion of subproperty and S ⊑ R, but the flawed property hierarchy (second column in Table 1) with S ⊑ R, S ⊑ ED × ED, and R ⊑ PED × PED results either in a classification of S's domain class (Ed_1) elsewhere in the taxonomy or it is inconsistent (column 3), depending on its original position in the class hierarchy. ♢

The observations are due primarily to currently implemented justification and explanation algorithms (e.g., [12,16,24]) that focus on logical deductions in OWL only and assume that class axioms and assertions about instances take precedence over what 'ought to be' regarding the object property expressions. 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 taken as fixed and assumed to be correct, but instances and classes can move about in the taxonomy when classifying the ontology. However, the modeller may be certain where in the taxonomy a particular class should be positioned, or at least its main category, but not sure about how to represent its properties. This is a reasonable assumption, given that many modellers commence ontology development by first adding a taxonomy and only gradually add properties, and the advanced OWL 2 DL features for object properties are relatively new and not easy to grasp. 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—i.e., preventing a flaw—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. This requires an ability to recognise the flaw, to explain it, and to suggest revisions, and based on that, attempt to prevent it.

Table 1 Sample ontologies to illustrate the OWL reasoners and explanation (based on Protégé's explanation feature) of defects supposedly in the class hierarchy, although the real modelling flaw is in the object property hierarchy.

$\mathcal{O}_1$	**OPEs**	**CEs**	**Deduced, with explanation**
	$\mathsf{R} \sqsubseteq \mathsf{PED} \times \mathsf{PED}$ $\mathsf{S} \sqsubseteq \mathsf{ED} \times \mathsf{ED}$ $\mathsf{S} \sqsubseteq \mathsf{R}$	OWLized Figure 2, $\mathsf{Ed}_1 \sqsubseteq \mathsf{ED}$, $\mathsf{Ed}_2 \sqsubseteq \mathsf{ED}$, $\mathsf{Ped}_1 \sqsubseteq \mathsf{PED}$, $\mathsf{Ped}_2 \sqsubseteq \mathsf{PED}$, $\mathsf{Ed}_1 \sqsubseteq \exists \mathsf{S}.\mathsf{Ed}_2$, $\mathsf{Ped}_1 \sqsubseteq \exists \mathsf{R}.\mathsf{Ped}_2$	$\mathsf{Ed}_1 \sqsubseteq \mathsf{PED}$: because the domain of R is PED and $\mathsf{S} \sqsubseteq \mathsf{R}$
$\mathcal{O}_2$	**OPEs**	**CEs**	**Deduced, with explanation**
	as $\mathcal{O}_1$	as $\mathcal{O}_1$, but with $\mathsf{Ed}_2 \sqsubseteq \mathsf{AS}$ (and $\mathsf{PED} \sqsubseteq \neg \mathsf{AS}$ still holds)	Ed_1 inconsistent: 1. $\mathsf{AS} \sqsubseteq \neg \mathsf{PED}$, 2. $\mathsf{Ed}_1 \sqsubseteq \exists \mathsf{S}.\mathsf{Ed}_2$, 3. $\mathsf{Ed}_2 \sqsubseteq \mathsf{AS}$, 4. R's range is PED, 5. $\mathsf{S} \sqsubseteq \mathsf{R}$

We address these issues by, first, defining two non-standard reasoning services. We extend the RBox Compatibility Service for object subproperties [18] to also handle the object property characteristics, called Sub-Property compatibility Service (*SubProS*), and we define a new reasoning service, the Property Chain compatibility Service, (*ProChainS*), that checks whether the chain's properties are compatible. The compatibility services are defined such that it exhaustively checks all permutations and therewith pinpoints to the root cause in the OWL object property expressions, where applicable. Second, if a test of either service fails, proposals are made to revise the identified flaw. As such, *SubProS* and *ProChainS* may be considered extra-logical or onto-logical 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 [28], isolated mistakes and suggestions [27], and the catalogue of common pitfalls [25], but then including also a systematic account of why the flaws occur and how to revise them, alike an OntoClean [7] for object property expressions. Hence, it is different from other works on explanation and pinpointing mistakes that concern logical consequences only [12,16,24], and *SubProS* and *ProChainS* also propose revisions for the flaws. In addition, these foundations also shed light on problems with distributivity of properties [1,15,26], and over parthood relations in particular, which can be used to prevent flaws.

In the remainder of the paper, we characterise property subsumption and define *SubProS* in Section 2, and property chaining with *ProChainS* in Section 3. Suggestions to avoid the need for *ProChainS* are discussed in Section 4. We conclude in Section 5.

2 Sub-Properties in OWL

After summarizing OWL object property expressions and its DL version of role inclusion axioms, we proceed to the notion of property subsumption thanks to domain and range axioms and thanks to a property's characteristics. This forms the basis for *SubProS*, which, in turn, is used to devise guidelines for how to revise each flaw, and subsequently is used in the evaluations with several ontologies.

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(`$\mathtt{OPE}_1$ $\mathtt{OPE}_2$`)`, which says that object property expression $\mathtt{OPE}_1$ is a subproperty of $\mathtt{OPE}_2$, 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" [23]. The simple version is denoted in Description Logics (DL) as $S \sqsubseteq R$ and a typical use case is $\mathsf{properPartOf} \sqsubseteq \mathsf{partOf}$. The more complex form concerns property chains, denoted with `SubObjectPropertyOf(ObjectPropertyChain(`$\mathtt{OPE}_1$ `...` $\mathtt{OPE}_n$`) OPE)` in OWL 2 where `OPE` is an object property expression, and several additional syntactic constraints hold: $n \geq 2$, and `OPE` is equal to `owl:topObjectProperty`, or $n = 2$ and $\mathtt{OPE}_1 = \mathtt{OPE}_2 = \mathtt{OPE}$, or $\mathtt{OPE}_i < \mathtt{OPE}$ for each $1 \leq i \leq n$, or $\mathtt{OPE}_1 = \mathtt{OPE}$ and $\mathtt{OPE}_i < \mathtt{OPE}$ for each $2 \leq i \leq n$, or $\mathtt{OPE}_n = \mathtt{OPE}$ and $\mathtt{OPE}_i < \mathtt{OPE}$ for each $1 \leq i \leq n-1$. This is called "complex role inclusions" in DL, and is defined succinctly in [13], which states that (sub-)properties are constrained by the Role Inclusion Axioms as defined in [13] for $\mathcal{SROIQ}$, the base language for OWL 2 DL, which also provides 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 (including where $i = 1$, hence the basic form), 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, provided the regularity constraint hold (a strict order) so as to maintain decidability.

Definition 1 ((Regular) Role Inclusion Axioms [13]) 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 $1 \leq i \leq n$, or
4. $w = R \circ S_1 \circ \ldots \circ S_n$ and $S_i \prec R$, for all $1 \leq i \leq n$, or
5. $w = S_1 \circ \ldots \circ S_n \circ R$ and $S_i \prec R$, for all $1 \leq i \leq n$.

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, henceforth, we will use this notation and "$\circ$" for chaining rather than the wordy OWL functional style syntax.

We look into the "basic form" for sub-properties, i.e., $S \sqsubseteq R$, in the remainder of this section and consider property chains in Section 3. To increase readability, we 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 (i.e., `ObjectPropertyDomain(OPE CE)` and `ObjectPropertyRange(OPE CE)` in OWL, respectively), and $R \sqsubseteq \top \times \top$ holds when no explicit domain and range axiom has been declared (in OWL: `ObjectPropertyDomain(R owl:Thing)` and `ObjectPropertyRange(R owl:Thing)`) unless it inherited a domain or range from its parent property.

2.2 When a property is a subproperty of another

It is well-known that for class subsumption the subclass has to be 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. Put differently: 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 principle ways to constrain a property, and either one suffices: (i) specify its domain and/or range, and (ii) declare the property's characteristics[3], which will be analysed in detail in the next two subsections.

[3] Note that a cardinality constraint applies to the axiom, not the property, hence, is not considered here.

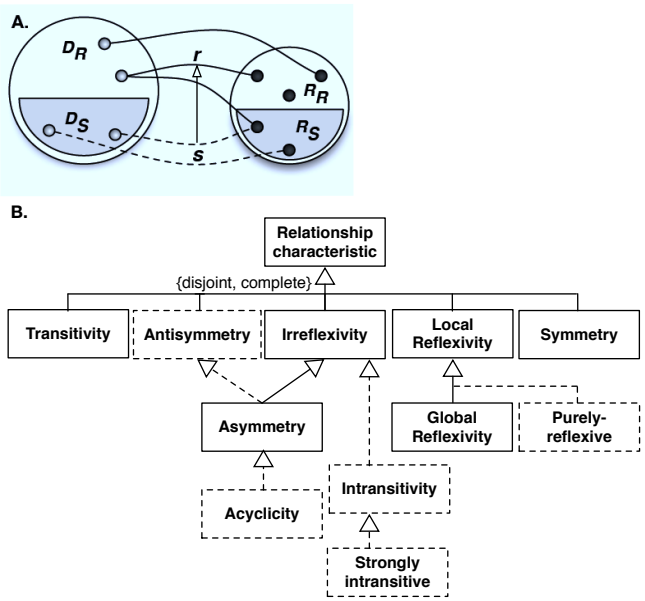

Fig. 3 A: Depiction of a structure satisfying $S \sqsubseteq R$ thanks to the domain and range restrictions ($D_S \sqsubseteq D_R$ and $R_S \sqsubseteq R_R$), alike the so-called 'subsetting' idea in UML; B: hierarchy of property characteristics (updated from [8,9]), where dashed characteristics cannot be represented in OWL 2 DL.

2.2.1 Subsumption due to domain and range axioms

Given an object 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. This might be perceived to have an object-oriented and conceptual data modelling flavour, yet it is widely used (see also Table 2 on the evaluation of ontologies) and declaring domain and range axioms has distinct advantages in automated reasoning as well as for constraining the admissible models and therewith being more precise in representing the knowledge. The intuition is depicted graphically with a possible model in Figure 3-A. To make this more precise, let us first introduce the notion of *user-defined domain and range classes* for OWL ontologies in Definition 2.

Definition 2 (User-defined Domain and Range Classes) Let R be an OWL object property, C_i an OWL class that may or may not be atomic and $1 \leq i \leq n$, 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., $R_R = C_2$.

Observe that one can have a D_R or R_R that is, e.g., the union of two classes, and if an object property has a user-defined domain and/or range declared,

then its subproperties will inherit the user-defined domain/range unless the subproperty has its own user-defined domain and/or range declared that overrides the inherited one.

Thus, for the property subsumption due to domain or range, we need to specify that it ought to be the case that, given an axiom $S \sqsubseteq R$, $D_S \sqsubseteq D_R$ and $R_S \sqsubseteq R_R$ must hold, 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—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—as was illustrated in Example 1. This kind of problem was addressed in [18] with the *RBox Compatibility* service, for which the DL $\mathcal{ALCI}$ sufficed to define it. We will adapt it to OWL 2 DL and substantially extend that service and options to correct it in Section 2.3.

2.2.2 Subsumption due to the property's characteristics

OWL object property characteristics (relational properties) constrain the way objects relate to each other; e.g., if an ABox contains $\mathsf{connectedTo}(\mathsf{a},\mathsf{b})$, then only if $\mathsf{connectedTo}$ is asserted to be symmetric then it will infer $\mathsf{connectedTo}(\mathsf{b},\mathsf{a})$. One can argue for a property hierarchy that respects 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 vice versa. Thus, this relies on how the property characteristics relate among themselves, which is depicted as a small hierarchy in Figure 3-B. Strongly intransitive and purely reflexive have been identified recently by [9]; for completeness and self-containedness we include the characterisations of all property characteristics in first order logic notation, as they are more generic and not all of them have been captured in a DL before:

- transitive: $\forall x,y,z(R(x,y) \wedge R(y,z) \rightarrow R(x,z))$;
- intransitive: $\forall x,y,z(R(x,y) \wedge R(y,z) \rightarrow \neg R(x,z))$;
- strongly intransitive: $\forall x,y,z((R(x,y) \wedge P(y,z) \rightarrow \neg R(x,z)$ and "P is recursively defined to give the transitive closure of R, i.e., $\forall x,y(P(x,y) \leftarrow (R(x,y) \vee \exists z(R(x,z) \wedge P(z,y))))$" so as to prohibit 'jumping' over one or more classes in the hierarchy [9];
- locally reflexive: $\forall x,y(R(x,y) \rightarrow R(x,x))$, which requires a y to be there before x relates to itself through R;
- globally reflexive: $\forall x R(x,x)$;
- purely reflexive: $\forall x,y(R(x,y) \rightarrow x = y)$ [9];
- irreflexive: $\forall x \neg R(x,x)$;
- asymmetric: $\forall x,y(R(x,y) \rightarrow \neg R(y,x))$;
- symmetric: $\forall x,y(R(x,y) \rightarrow R(y,x))$;
- antisymmetric: $\forall x,y(R(x,y) \wedge R(y,x) \rightarrow x = y))$, or, as in [8]: $\forall x,y(\neg(x = y) \wedge R(x,y) \rightarrow \neg R(y,x))$;
- acyclic: R is acyclic iff $\forall x \neg(x$ has path to $x)$ [8].

A subproperty has to be more constrained than its parent property and, as with inheritance of properties for classes, a property's property should be inherited along the property hierarchy and overridden with a stronger constraint if that is declared on the subproperty. With this line of reasoning and $S \sqsubseteq R$, then $\mathsf{Asym}(S)$ and $\mathsf{Irr}(R)$ is acceptable, 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 relation and its sub-relation proper parthood: the former is reflexive, antisymmetric, and transitive, and the latter irreflexive, asymmetric, and transitive. With parthood and transitivity, however, there is a complicating factor. Direct parthood ($\mathsf{direct\text{-}part}$) is intransitive and tends to be represented in an OWL ontology as a subproperty of parthood ($\mathsf{part\text{-}of}$)[4]. The problem is that one cannot represent intransitivity in OWL explicitly and not asserting transitivity means a property is non-transitive, not intransitive. Informal feedback from the community revealed that inheritance of transitivity along the object property hierarchy is contentious, where either it is assumed this is definitely the case and should hold, or that it definitely must not be inherited down in the property hierarchy and if it were a deduction by an automated reasoner, it would be a bug. At present, in Protégé 4.1 (Mac version, build 239) with the HermiT v1.3.6 OWL 2 reasoner as well as with FaCT++ v1.6.0, transitivity is not inherited but instead passed on upward in the hierarchy at the instance-level (OWL ABox), which has certain consequences for the design of *SubProS* as well as modelling guidelines on prevention of 'unexpected' deductions. We illustrate some consequences first in the following example, considering transitive, non-transitive, and pretended-to-be intransitive parthood relations in an OWL ontology.

Example 2 (Transitivity in OWL ontologies) We use the African Wildlife Ontology to demonstrate certain, perhaps surprising, deductions. We have a defined class

$$\mathsf{Herbivore} \equiv \exists \mathsf{eats}.(\mathsf{Plant} \sqcup \exists \mathsf{is\text{-}part\text{-}of}.\mathsf{Plant}) \sqcap \forall \mathsf{eats}.(\mathsf{Plant} \sqcup \exists \mathsf{is\text{-}part\text{-}of}.\mathsf{Plant}),$$

$\mathsf{Giraffe}$ is asserted as a subclass of Animal and

[4] Two informal, but well-known, sources promoting this representation are: the W3C Best Practices document on Simple Part-Whole relations, `http://www.w3.org/2001/sw/BestPractices/OEP/SimplePartWhole/`, and the Componency Ontology Design Pattern at `http://ontologydesignpatterns.org/wiki/Submissions:Componency`.

Giraffe $\sqsubseteq \exists$eats.(Leaf $\sqcup$ Twig) $\sqcap \forall$eats.(Leaf $\sqcup$ Twig)
For the current purpose, we use direct parthood, where Leaf is a direct-part of a Twig or Branch, Twig is a direct-part of a Branch, and a Branch is a direct-part of a Tree that in turn is a subclass of Plant, and, as usual, Trans(is-part-of). This being the same, we create two ontologies such that `AWOsister` has direct-part as *sibling* object property of is-part-of and `AWOsub` has direct-part as *sub*-object property of is-part-of, and neither ontology has any characteristics (relational properties) checked for direct-part.

After synchronizing the reasoner (be this HermiT or Fact++), Giraffe is still a direct subclass of Animal in `AWOsister`, whereas we do obtain the deduction that Giraffe is a Herbivore in `AWOsub`. The reason for this difference is, that in `AWOsub` that what is related through direct-part is also related through its transitive parent property is-part-of, and therefore leaves and twigs are part of plants by is-part-of's transitivity, and therefore giraffe is classified as a herbivore.

If, on the other hand, we change the definition of Herbivore by replacing the is-part-of occurrences with direct-part, then giraffe is *not* classified as a herbivore, because direct-part is not explicitly declared transitive. This as-if behaviour for intransitivity does not work well, however, once we add is-proper-part-of $\sqsubseteq$ direct-part and Trans(is-proper-part-of). Then with individuals and their relations is-proper-part-of(aleaf1, atwig1) and is-proper-part-of(atwig1, atree1), we obtain the expected deduction that is-proper-part-of(aleaf1, atree1), but also direct-part(aleaf1, atwig1), direct-part(atwig1, atree1), and direct-part(aleaf1, atree1), and likewise for is-part-of. In other words: direct-part starts behaving just like if it were transitive, contrary to the modeller's intention. $\diamondsuit$

The quasi-nonmonotonicity illustrated in the example is even more undesirable compared to simply being consistent in positioning a nontransitive, but desired to be intransitive, property at the top-level of the property hierarchy and annotating it that one should not add a transitive subproperty.

Overall, however, subsumption in the object property hierarchy due to the properties' characteristics is fairly straight-forward for OWL 2 DL, because antisymmetry, acyclicity, and intransitivity cannot be represented, nor the strongly intransitive and purely reflexive characteristics. Current OWL reasoners do not take into account inheritance of property characteristics, but instead a characteristic is distributed upwards for individuals; that is, if, say, Sym(R) then this does not hold automatically for S (with $S \sqsubseteq R$), but if Sym(S) then all objects that relate through S symmetrically also do so symetrically through R, even though it was not explicitly stated as such. Also with symmetry the deductions exhibit a quasi-nonmonotonicity of the super-property depending on the assertions on its subproperty, alike what we have seen for non-transitivity. To be sure, adding Asym(S) is very well possible, and the argument holds likewise for Ref and Irr.

An avenue to remedy this, is to add something to the reasoners, or one can design a 'pre-reasoner check' to alert the ontologist to such a case and propose to add it or change the object property characteristics expression(s). Either way, a general principle holds as to how the property hierarchy should behave, which we shall define in the next section.

2.3 The SubProperty compatibility Service

Given the analysis on subproperties, we now can define the new reasoning service, *SubProperty compatibility Service* (*SubProS*), which we do by extending the basic notions from the *RBox compatibility* [18]. 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 declared for R.

Definition 3 (SubProperty 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 [23], 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 $\mathcal{O} \models$ Asym(R) then $\mathcal{O} \models$ Asym(S);
Test 5. If $\mathcal{O} \models$ Sym(R) then $\mathcal{O} \models$ Sym(S) or $\mathcal{O} \models$ Asym(S);
Test 6. If $\mathcal{O} \models$ Trans(R) then $\mathcal{O} \models$ Trans(S);
Test 7. If $\mathcal{O} \models$ Ref(R) then $\mathcal{O} \models$ Ref(S) or $\mathcal{O} \models$ Irr(S);
Test 8. If $\mathcal{O} \models$ Irr(R) then $\mathcal{O} \models$ Irr(S) or $\mathcal{O} \models$ Asym(S);
Test 9. If $\mathcal{O} \models$ Asym(R) then $\mathcal{O} \not\models$ Sym(S);
Test 10. If $\mathcal{O} \models$ Irr(R) then $\mathcal{O} \not\models$ Ref(S);
Test 11. If $\mathcal{O} \models$ Trans(R) then $\mathcal{O} \not\models$ Irr(R), $\mathcal{O} \not\models$ Asym(R), $\mathcal{O} \not\models$ Irr(S), and $\mathcal{O} \not\models$ 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*. Practically, several *SubProS* tests can avail in part of the extant OWL reasoners. Class subsumption checking for `Tests 1-3` can be done with a call to the API of 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, given that 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.

2.3.1 Remedying a flaw

Assuming a modeller may want to remedy a detected flaw, we specify what has to be done if any of the applicable tests fails. For `Tests 1-3`, we reuse the basic 2-step idea from [18] and adapt it to the current setting, and we propose new corrections for `Tests 4-11`. Concerning terminology, "raising a warning" denotes that it is not a logical error with respect to the current standard OWL automated reasoners but an extra-logical or onto-logical one, "forcing" a revision indicates that the flaw resulted also in 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 possible corrections:
   - ⋆ Change the object property hierarchy by either removing $S \sqsubseteq R$ and adding $R \sqsubseteq S$ or adding $S \equiv R$ 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 is a new class such that $D'_R \equiv D_R \sqcap D_S$, 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 possible corrections:
   - ⋆ 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 (which will 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 ex-

pression" and force the modeller to change it by either:
  - ⋆ Remove Trans(R), or
  - ⋆ Remove Irr(R), Asym(R), Irr(S), and Asym(S);

I. Run *SubProS* again if any changes have been made in steps D-H, and check any changes made in the property hierarchy 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 it must be run again to both push through the changes and prevent an infinite loop.

2.3.2 Evaluation of SubProS

SubProS was evaluated with several ontologies. The TONES Ontology repository[5] contains 219 ontologies and we selected 12 ontologies semi-randomly based on (i) having listed in the metrics to have a substantial amount of object properties, and (ii) being a real ontology (i.e., no toy or tutorial ontology, not converted from OBO, nor an OWLized thesaurus). Relevant data about the ontologies and the outcomes of running *SubProS* is shown in Table 2. Noteworthy is that object properties that are in an hierarchy predominantly have changes in the domain or range. We analyse BioTop's inconsistent object property in the next evaluation.

Evaluation 1 (BioTop's inconsistent 'has process role') The 'has process role' in BioTop [2] version d.d. June 17, 2010 is inconsistent, yet there is no explanation for it computed by currently implemented explanation algorithms. The ontology contains, among other axioms:

'has process role' $\sqsubseteq$ 'temporally related to' (E.1)
'has process role' $\sqsubseteq$ 'processual entity' $\times$ role (E.2)
'temporally related to' $\sqsubseteq$ ('processual entity' $\sqcup$ quality $\times$ 'processual entity' $\sqcup$ quality) (E.3)
role $\sqsubseteq \neg$quality (E.4)
role $\sqsubseteq \neg$'processual entity' (E.5)
Sym('temporally related to') (E.6)

We now use *SubProS* to isolate the flaw and subsequently propose a revision.

- `Test 1`: fail, because $R_{\mathsf{hasprocessrole}} \sqsubseteq R_{\mathsf{temporallyrelatedto}}$ is false, as the ranges (see E.2 cf. E.3) are disjoint (see E.4, E.5) and therewith 'has process role' is inconsistent;
- `Test 2` and `3`: pass.
- `Test 4`: not applicable.
- `Test 5`: fail, because $\mathcal{O}$ does not contain Sym('has process role').
- `Test 6-11`: not applicable.

To revise the issue detected with `Test 1`, we have to choose among three options, as described in item `A`, above, regarding remedying a flaw. The 'has process role' property has not been used in class expressions, and from the informal meaning described in the annotation, it fits 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, and the same holds for the inverse ('process role of'.). To revise the issue of the failed `Test 5`, there are three options (see item `E`): it was already decided not to change the property hierarchy, so one either can add Sym('has process role'), or Asym('has process role'), or remove Sym('temporally related to'). Considering the meaning of the properties, Sym('temporally related to') is certainly true and Sym('has process role') certainly false, hence Asym('has process role') is the appropriate revision. After changing the ontology accordingly, no issues emerge with *SubProS*. ♢

[5] `http://owl.cs.manchester.ac.uk/repository/`

3 Property Chaining in OWL

Property chaining is well-known in DL research as role composition [22,29,33], but in OWL its usage is more restricted and, except for a few elaborate cases (the family relations example [23] and metamodelling rewritings [6]), is typically used with only two object properties being chained, where one of them occurs on both sides of the inclusion [4,13,23]; e.g., hasPart $\circ$ hasParticipant $\sqsubseteq$ hasParticipant in pharmacogenomics [5], foaf:maker $\circ$ foaf:maker$^-$ $\sqsubseteq$ sda:co-author in the conferencing domain [3], and similarly for digital objects collections (dcterms:contributor$^-$ $\circ$ dcterms:contributor $\sqsubseteq$ ex:co-author) [20]. We will first describe some background information about property chaining and its 'original', role composition, illustrate some typical problems, then introduce the Property Chain compatibility Service *ProChainS*, and finally address how to manage consequences of property chains and revise flawed ones. The issues mentioned and solution proposed here applies to any OWL 2 DL admissible chain.

3.1 Preliminaries and problems

Property chaining in OWL 2 has a long history in DL research, where it is known as *role composition* [14,22,

Table 2 Selection of some TONES Repository ontologies, retrieved on 12-3-2012, and others used in the examples; OP = object property; char. = object property characteristic.

Ontology	No. of OPs	No. of SubOP axioms	No. more constrained		Comments
			D or R	by char.	
DOLCE-lite	70	46	13	3	Transitivity was added to a subproperty.
gfo-basic	41	12	11	1	Functionality was added to a subproperty.
SAO 1.2	36	25	21	5	Additions to a subproperty cf. its parent property: 2 instances where functionality is added, and 2 x transitivity added; `Test 6` of *SubProS* fails on has Vesicle Component.
airsystem	111	56	43	2	One subproperty has functionality added; *ProChainS*'s `Test SR-a`, `b` and `c` fails on hasFunctionalSubsystem ∘ hasComponent ⊑ hasComponent because hasFunctionalSubsystem does not have user-defined domain and range axioms, whereas hasComponent has Object.
process (SWEET)	102	10	7	0	Mainly many underspecifications and implicit imports (e.g., space.owl), which hampers analysis.
family-tree	52	25	14	2	Functionality is added to one subproperty; fails `Test 6` of *SubProS*.
propreo	32	20	17	2	The ontology is beyond OWL 2 DL because a non-simple object property appears in a class axiom that uses a maximum qualified cardinality constraint.
heart	29	18	9	0	There are many inconsistencies; fails `Test 1` and `2` on isBranchOf; fails `Test 6` of *SubProS* in many cases (including the direct-part issue, with, e.g., has direct subcavity).
mygrid-unclassified	69	39	0	3	Transitivity was added once to a subproperty and functionality twice.
building architecture	28	24	0	0	It fails `Test 5` of *SubProS* due to the omission of Asym on properPartOf.
biotop	89	84	45	9	Transitivity was added to two subproperties; fails `Test 1` and `5` of *SubProS*, which is discussed in *Evaluation 1*.
legal-action.owl					The ontology did not load in Protégé due to multiple dependencies, and timed out.
biochemical-reaction-complex	7	0	0	1	It contains the RO relations and the untyped property chain with hasparticipant and haspart, which is discussed in *Evaluation 4*.
DMOP v5.2	118	40	23	1	Functionality is added to a subproperty; fails `Test 6` of *SubProS*; analyses of DMOP with *ProChainS* are described in *Example 4*, and *Evaluation 2* and *5*.
family-tree	52	25	12	0	It fails `Test 5` and `6` of *SubProS*, including the hasAncestor and hasGrandparent issue.

29,33]. There are subtle differences between the original notion of composition in DL and what remains of it in OWL such that, on the one hand, the the effective meaning of a property chain in OWL is considerably weaker than the true role composition, and, on the other hand, leaves room for 'workarounds'. We briefly summarise and discuss the main results on role composition (without loss of generality, we illustrate it with only two roles on the left-hand side of the inclusion). This aids examining more precisely correct and flawed property chain axioms.

Role composition in the DL language $\mathcal{ALR}$ is defined as (Eq. 1) [29],

$$R \circ S \doteq \{(x,y) | \exists z \in M : (x,z) \in R \wedge (z,y) \in S\} \quad (1)$$

where R and S are DL-roles, i.e., OWL object properties, over a set M, that is $R, S \subseteq M \times M$. The composition "$R \circ S$" means that the two instances x and y are related by *some* relation—unnamed in (Eq. 1)—which amounts to the same as joining the two roles. Logically, and ontologically, this is "true composition". Let us name the implicit or derived DL-role indicated with the right-hand side of the "$\doteq$" as T, hence, $R \circ S \doteq T$, where R, S and T may be the same or different; e.g., child ∘ child $\doteq$ grandchild, i.e., the child of your child is your grandchild. This true composition with arbitrary roles is undecidable and therewith undesirable from a computational viewpoint, and so is its weaker version $R \circ S \sqsubseteq T$ [29,33]. With two additional weakening steps limiting its usage, it is decidable and included in $\mathcal{RIQ}$ [14] and $\mathcal{SROIQ}$ [13]: one can represent only $R \circ S \sqsubseteq R$ or $S \circ R \sqsubseteq R$, where R and S may be the same or different and one has acyclic role hierarchy. For $\mathcal{SROIQ}$, the composition operator and its constraints for property chaining are declared as specified in Definition 1.

Although weakening of true role composition to inclusion is understandable from the viewpoint of the desire to stay within the decidable fragment of first order logic, practically, the restrictions on usage change the semantics and intention of 'composition'—defining a role from other roles—into mere 'chaining'—some limited necessary condition for that role. On the other hand, it is also underspecified from a purely modelling viewpoint, because it is possible to introduce behaviour as if it were composition, using a 'workaround' through under- or non-specification of the domain and range of the roles. This is illustrated in the next example, where we also show that a more precise formalization actually will detect the issue, which, however, points in the logically correct but ontologically wrong direction.

Example 3 (Simulating role composition) Let us take $R \circ S \sqsubseteq R$, and for explanatory purpose and brevity, denote with R^l the R on the left-hand side of the inclusion, and R^r the R on the right-hand side, i.e., $R^l \circ S \sqsubseteq R^r$. We can make the domain and range of R^l distinct from that of R^r without declaring it explicitly. For instance, we want to know which kind of amino acids are part of which enzyme(s). Let $R = \mathsf{structPartOf}$, intended to mean to relate parts and wholes that are types of material continuants and $S = \mathsf{hasFunction}$ that relates a material continuant to a function or role (in, e.g., the BFO foundational ontology), AminoAcid and Protein are both material continuants and Enzyme a function, and we have the following axioms in our ontology:

$$\mathsf{AminoAcid} \sqsubseteq \exists\ \mathsf{structPartOf.Protein} \quad \text{(E.1)}$$
$$\mathsf{Protein} \sqsubseteq \exists\ \mathsf{hasFunction.Enzyme} \quad \text{(E.2)}$$
$$\mathsf{structPartOf} \circ \mathsf{hasFunction} \sqsubseteq \mathsf{structPartOf} \quad \text{(E.3)}$$

This entails that

$$\mathsf{AminoAcid} \sqsubseteq \exists \mathsf{structPartOf.Enzyme}. \quad \text{(E.4)}$$

Deducing that an amino acid is part of some enzyme (E.4), is fine in the vernacular and used as such, but note the change in use and meaning of the structPartOf relation, which does not relate two material entities anymore as it should do, but now permits that a structural entity (AminoAcid) is part of a functional entity (Enzyme). That is, essentially, the high-level category of the range of structPartOf that is part of the definition of the meaning of the property has changed due to property chaining, and therewith the *intended meaning of* structPartOf *has changed*. If, regarding the intended meaning of R, we obtain $R^l \neq R^r$ resulting from the different implicit categories of the domain and range caused by the concatenation—even though the *name* of the role is exactly the same—we have introduced in disguise through the backdoor the $R \circ S \sqsubseteq T$, where $T \equiv R^r$. However, *this occurs only when domain or range axioms are either not or under-specified*. Let us now type the properties, where IC is IndependentContinuant and DC DependentContinuant:

$$\mathsf{structPartOf} \sqsubseteq \mathsf{IC} \times \mathsf{IC} \quad \text{(E.5)}$$
$$\mathsf{Function} \sqsubseteq \mathsf{DC} \quad \text{(E.6)}$$
$$\mathsf{hasFunction} \sqsubseteq \mathsf{IC} \times \mathsf{Function} \quad \text{(E.7)}$$
$$\mathsf{DC} \sqsubseteq \neg \mathsf{IC} \quad \text{(E.8)}$$

with the classes taken from BFO, and property chain (E.3), then we cannot instantiate it, because the ranges of structPartOf and hasFunction are disjoint (cf. E.5 and E.7, and E.8); hence, our approach was wrong. In addition, (E.1-E.8) taken together results in an unsatisfiable AminoAcid, which, however, is not the root cause of the problem. On the other hand, a chain like

$$\mathsf{structPartOf} \circ \mathsf{hasFunction} \sqsubseteq \mathsf{hasFunction} \quad \text{(E.9)}$$

implies that

$$\mathsf{AminoAcid} \sqsubseteq \exists\ \mathsf{hasFunction.Enzyme} \quad \text{(E.10)}$$

which leaves the semantics of the object properties intact and will not cause an inconsistency, but, unfortunately, this is ontologically nonsense: in this case, the part does not have the same function as the whole it is part of. Given the more precise knowledge represented, and still wanting to know the answer to the original question, one can obtain this by going in the other direction:

$$\mathsf{hasFunction}^- \circ \mathsf{structPartOf}^- \sqsubseteq \mathsf{hasFunction}^- \quad \text{(E.11)}$$

so that Enzyme inheres in (i.e., $\mathsf{hasFunction}^-$) some AminoAcids. ♢

Rephrasing the example: the workaround was incorrect from an ontology viewpoint and the vernacular was imprecise and therewith the motivation for the workaround was unsound, and being more precise in the representation of the knowledge resolved the issue[6]. Before we present a systematic approach to resolving issues with violating the meaning of roles/object properties, a different though related problem will be introduced, which is similar in flavour as Example 1 in illustrating undesirable deductions, but then applied to property chains. We have again two ontologies, $\mathcal{O}_1$ and $\mathcal{O}_2$, with their respective axioms as listed in Table 3 where the property chain is one of *Case S* from Definition 1. A standard reasoner deduces either a classification of a class elsewhere in the class hierarchy or an inconsistency in the class hierarchy, which is, in fact, due to a problematic property chain axiom together with the domain and range declarations of its participating properties. That this is not a contrived situation is demonstrated in Example 4 in the domain of data mining.

Example 4 (DMOP ontology chaining issue) The Data Mining and OPtimisation ontology (DMOP, v5.2, Sept.

[6] Further, as might be gleaned from (E.11), it is better to have (E.1) the other way around (Protein $\sqsubseteq \exists$ hasPart.AminoAcid), because not all amino acids are part of a protein, yet each protein must have some amino acids.

Table 3 Sample ontologies (CEs include the OWLized Fig 2) and a problematic property chain, with similar outcomes as in Table 1 but a more elaborate root problem (explanation based on Protégé 4's explanation feature).

$\mathcal{O}_1$	**OPEs**	**CEs**	**Inferred, with explanation**
	$\mathsf{R} \sqsubseteq \mathsf{POB} \times \mathsf{POB}$ $\mathsf{S1} \sqsubseteq \mathsf{PT} \times \mathsf{PT}$ $\mathsf{S2} \sqsubseteq \mathsf{ED} \times \mathsf{ED}$ $\mathsf{S3} \sqsubseteq \mathsf{PED} \times \mathsf{PED}$ $\mathsf{S1} \circ \mathsf{S2} \circ \mathsf{S3} \sqsubseteq \mathsf{R}$	$\mathsf{Pob}_1 \sqsubseteq \mathsf{POB}$, $\mathsf{Pob}_2 \sqsubseteq \mathsf{POB}$, $\mathsf{Ed}_1 \sqsubseteq \mathsf{ED}$, $\mathsf{Ed}_2 \sqsubseteq \mathsf{ED}$, $\mathsf{Pt}_1 \sqsubseteq \mathsf{PT}$, $\mathsf{Pt}_2 \sqsubseteq \mathsf{PT}$, $\mathsf{Pt}_1 \sqsubseteq \exists \mathsf{S1}.\mathsf{Pt}_2$, $\mathsf{Pt}_2 \sqsubseteq \exists \mathsf{S2}.\mathsf{Ed}_1$, $\mathsf{Ed}_1 \sqsubseteq \exists \mathsf{S3}.\mathsf{Pob}_1$	$\mathsf{Pt}_2 \sqsubseteq \mathsf{ED}$ due to domain S2, $\mathsf{Ed}_1 \sqsubseteq \mathsf{PED}$ due to domain S3, $\mathsf{Pt}_1 \sqsubseteq \mathsf{POB}$ due to the property chain
$\mathcal{O}_2$	**OPEs**	**CEs**	**Inferred, with explanation**
	as $\mathcal{O}_1$	as $\mathcal{O}_1$, but with $\mathsf{Pt}_1 \sqsubseteq \mathsf{M}$ (note: $\mathsf{ED} \sqsubseteq \neg\mathsf{PD}$ still holds)	Pt_1 inconsistent because 1. $\mathsf{M} \sqsubseteq \neg\mathsf{POB}$, 2. $\mathsf{Ed}_1 \sqsubseteq \exists \mathsf{S3}.\mathsf{Pob}_1$, 3. $\mathsf{Pt}_1 \sqsubseteq \mathsf{M}$, 4. $\mathsf{Pt}_2 \sqsubseteq \exists \mathsf{S2}.\mathsf{Ed}_1$, 5. $\mathsf{D_R} = \mathsf{POB}$, 6. $\mathsf{S1} \circ \mathsf{S2} \circ \mathsf{S3} \sqsubseteq \mathsf{R}$

2011) was developed in the e-LICO project (`http://www.e-lico.eu`) [11]. It uses the $\mathcal{SROIQ}$ features, has some 573 classes, 1021 subclass axioms, 113 object properties, 40 object sub-property axioms, and 11 property chains. One of the chains, shown in Figure 4, is

$$\mathsf{hasMainTable} \circ \mathsf{hasFeature} \sqsubseteq \mathsf{hasFeature}, \quad \text{(E.1)}$$

and the properties are typed as

$$\mathsf{hasMainTable} \sqsubseteq \mathsf{DataSet} \times \mathsf{DataTable}, \quad \text{(E.2)}$$

$$\mathsf{hasFeature} \sqsubseteq \mathsf{DataTable} \times \mathsf{Feature}. \quad \text{(E.3)}$$

Observe that $R_{\mathsf{hasMainTable}} = D_{\mathsf{hasFeature}}$—both are DataTable—but $D_{\mathsf{hasMainTable}} = \mathsf{DataSet}$ and $D_{\mathsf{hasFeature}} = \mathsf{DataTable}$. Because DataSet is a not-disjoint sibling-class of DataTable in DMOP, the combination of (E.2, E.3) and (E.1) causes the deduction $\mathsf{DataSet} \sqsubseteq \mathsf{DataTable}$. This deduction is wrong with respect to the subject domain semantics, and the real flaw is either a domain or range axiom, or there is a flaw in the chain axiom. We solve this issue with *ProChainS* in Evaluation 5. ♢

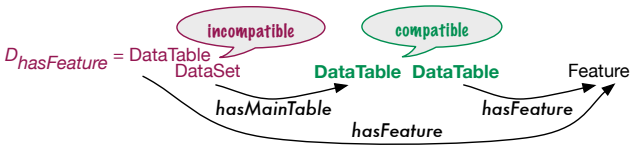

Fig. 4 Graphical depiction of $\mathsf{hasMainTable} \circ \mathsf{hasFeature} \sqsubseteq \mathsf{hasFeature}$, the domain and range axioms of the two object properties, and compatibility of the domains and ranges.

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 a specification of new constraints on the use of property chains so as to guarantee meaningful reasoning with respect to the subject domain, and explanation of the derivations, which can be implemented in a reasoner and/or in the interface of the ontology development software. We address the issues with the *Property Chain compatibility Service* (*ProChainS*), which ensures that 'safe' property chains are declared in the ontology.

3.2 The Property Chain Compatibility Service

With respect to Definition 1's 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 S*, i.e., the third option in Definition 1, 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. Informally, to ensure avoidance of undesirable classifications or inconsistencies, the domain/range class from left to right has to be equal or a superclass on the left-hand side of the inclusion, and similarly for the outer domain and range on the left-hand side and domain and range of the object property on the right-hand side.

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 [23]) 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 [23]), 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 evaluated with three domain ontologies that, for ease of explanation, we assume to contain an OWLized DOLCE taxonomy (recall Figure 2) where applicable, it has subject domain classes (e.g. DrugTreatment) that are involved in class expressions, DOLCE's hasParticipant $\sqsubseteq$ PD $\times$ ED and hasPart $\sqsubseteq$ PT $\times$ PT, a structuralPart $\sqsubseteq$ POB $\times$ POB, and a part-of between perdurants (involvedIn $\sqsubseteq$ PD $\times$ PD).

Evaluation 2 (DMOP chains, Case S) DMOP v5.2 contains a property chain realizes $\circ$ addresses $\sqsubseteq$ achieves and each property has a domain and range axiom, which is depicted in Figure 5; the classes are all subclasses of DOLCE's NPED. Let us apply the three relevant tests of *ProChainS*:

- `Test S-a`: pass, because $R_{\mathsf{realizes}} \sqsubseteq D_{\mathsf{addresses}}$;
- `Test S-b`: pass, because the answer to "$D_{\mathsf{realizes}} \sqsubseteq D_{\mathsf{achieves}}$?" is that they are both DM-Operation;
- `Test S-c`: fail, because $R_{\mathsf{achieves}} \sqsubseteq R_{\mathsf{addresses}}$ holds instead of $R_{\mathsf{addresses}} \sqsubseteq R_{\mathsf{achieves}}$; that is, $R_{\mathsf{addresses}}$ is the *union* of DM-Task and OptimizationProblem, whereas R_{achieves} is only DM-Task.

Notwithstanding the failed `Test S-c`, the chain can be instantiated, but if the range of addresses is a subclass of OptimizationProblem in a class expression, then its instances will be classified as a member of DM-Task, 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 (due to other axioms, the root problem was identified as a "bad individual" member of DM-Operation), instead of pointing to the issue with $R_{\mathsf{addresses}}$ and R_{achieves} and the property chain. Furthermore, the classification is undesirable: tasks and problems are clearly distinct entities. The lead ontology developer chose to revise the domain and range restrictions of addresses to have the chain functioning as intended (included in v5.3). $\diamondsuit$

Evaluation 3 (SNOMED CT's injury, Case RS) Let us revisit SNOMED-CT from the introduction and the so-called upwards distributivity, which was asserted it should hold for injuries. First, add to $\mathcal{O}$ the chain

injuryOf $\circ$ structuralPart $\sqsubseteq$ injuryOf

so that when some Fracture $\sqsubseteq$ F as an injury of Metacarpal2 $\sqsubseteq$ POB—which is the bone between the wrist and the index finger, hence, a structural part of the Hand—then the reasoner infers it is also an injury of the hand. Running *ProChainS*, we obtain:

- `Test RS-a`: pass, for $i = 1$;
- `Test RS-b`: pass, because "$R_{\mathsf{injuryOf}} \sqsubseteq D_{\mathsf{structuralPart}}$?" has all POBs;
- `Test RS-c`: pass, because "$R_{\mathsf{structuralPart}} \sqsubseteq R_{\mathsf{injuryOf}}$?" has all POBs.

Hence, according to *ProChainS*, it is a compatible property chain.$\diamondsuit$

Evaluation 4 (Pharmacogenomics chains, Case SR) The pharmacogenomics ontology contains the chain hasPart $\circ$ hasParticipant $\sqsubseteq$ hasParticipant and knowledge about drugs and treatments [5], and aforementioned axioms. Evaluating it with *ProChainS*, we obtain:

- `Test SR-a`: pass, as it is trivially satisfied ($i = 1$);
- `Test SR-b`: fail, because $D_{\mathsf{hasPart}} \sqsubseteq D_{\mathsf{hasParticipant}}$ does not hold, because PD $\sqsubseteq$ PT;
- `Test SR-c`: fail, because $R_{\mathsf{hasPart}} \sqsubseteq D_{\mathsf{hasParticipant}}$ does not hold, because PD $\sqsubseteq$ PT.

Extending $\mathcal{O}$ with DrugTreatment $\sqsubseteq$ PT and DrugGeneInteraction $\sqsubseteq$ PT, then a reasoner deduces they are subclasses of PD, whereas if DrugTreatment $\sqsubseteq$ ED and DrugGeneInteraction $\sqsubseteq$ ED were to be added to $\mathcal{O}$, then it deduces that the two classes are inconsistent because ED $\sqsubseteq \neg$PD. The usage of "hasPart" thus holds only if DrugTreatment and DrugGeneInteraction are subclasses of PD (perdurants or 'processes'). This observation permits one to refine the property chain into

involvedIn$^-$ $\circ$ hasParticipant $\sqsubseteq$ hasParticipant.

With this refinement, we run *ProChainS* for *Case SR* again: `Test SR-b` and `c` pass, because $D_{\mathsf{involvedIn}^-} = R_{\mathsf{involvedIn}^-} = D_{\mathsf{hasParticipant}} = \mathsf{PD}$; hence, this property chain is *guaranteed* not to lead to an inconsistency when the object properties are used in OWL axioms. $\diamondsuit$

Having seen how the three cases in *ProChainS* function, we now proceed to managing the consequences in a structured way.

3.3 Managing Consequences of Property Chains

As Evaluation 4 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 always 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,

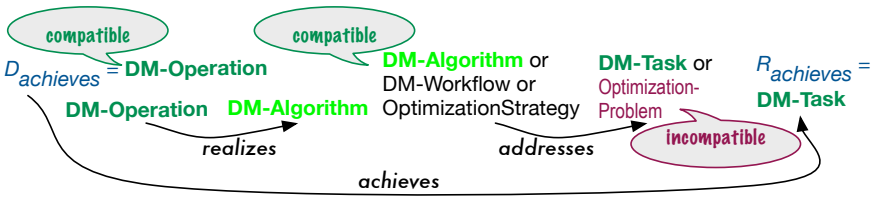

Fig. 5 The realizes ∘ addresses ⊑ achieves chain in DMOP v5.2, with the domain and range axioms of the participating properties; the matching chain is indicated in bold grey/green, the problematic one in Arial narrow font, and both are indicated with compatible or incompatible, respectively.

and suggest options to correct it in a similar way as for *SubProS*. A less comprehensive approach can be taken compared to the formal foundations of computing explanations or root justifications [12,24], because we do not have to find the root cause anymore and, in fact, can make use of certain explanation features that are implemented already. We propose the following ontology revisions, in case the test fails, where subscript i, with $i \leq n-1$, denotes the ordered property on the left-hand side of the inclusion:

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 an 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, solving the flaw described in Example 4.

Evaluation 5 (*Assessing DMOP chains*) DMOP v5.2 has 11 chains, of which eight raise a warning with *ProChainS*, and three of those cause a classification of classes elsewhere in the taxonomy due to the chain expressions. $\mathsf{hasMainTable} \circ \mathsf{hasFeature} \sqsubseteq \mathsf{hasFeature}$ was intorduced in Example 4, which is an instance of *Case SR*. Applying the three relevant tests, we obtain:

- `Test SR-a`: pass, for $i = 1$;
- `Test SR-b`: fail, due to the incompatible the domain axioms, as DataSet is a not-disjoint sibling-class of 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 for revision are either to change the domain or to accept the new classification. The DMOP domain experts chose the first option and changed the domain of hasFeature into $\mathsf{DataSet} \sqcup \mathsf{DataTable}$, which is included in DMOP v5.3.
- `Test SR-c`: pass, as both are DataTable.

No inconsistencies or unexpected classifications were deduced with the other five chains, principally thanks to the absence of disjointness axioms. For instance, the *Case S* described in Evaluation 2 with $\mathsf{realizes} \circ \mathsf{addresses} \sqsubseteq \mathsf{achieves}$ where `Test S-c` failed because $\mathcal{O} \not\models R_{\mathsf{DM\text{-}Task} \sqcup \mathsf{OptimizationProblem}} \sqsubseteq R_{\mathsf{DM\text{-}Task}}$. Considering the suggestions for revision, step B's first option to revise the ontology was chosen, i.e., removing OptimizationProblem from the range axiom of addresses, which is included as such in DMOP v5.3. This was chosen principally because the meaning of the relation was overloaded, and OptimizationStrategy was removed from the domain axiom, i.e., the optimization aspects were removed entirely from addresses. ♢

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 OWL object property expressions, being the property hierarchies, domain and range axioms to type the property, a property's characteristics, and property chains.

4 Prospects for preventing flaws in property chains

Property chaining is a relatively recent addition to OWL that intends to meet ontologists' requests, such as the recurring example where one wants to infer that a fracture of the neck of the femur is a fracture of the femur [15] and car engine ownership [13], and it is gradually being used in ontologies and its usage likely will increase. Although by definition, the logic allows the modeller to chain any two relations as long as they do not violate the OWL 2 limitations, in practice, a certain subset of all possible relations is used more often than others, and most combinations of chaining generic relationships are actually flawed. We discuss and demonstrate validity of the latter assertion by considering object properties in DOLCE, GFO, and RO in Section 4.1. The interesting chains that yield ontologically correct deductions concern plain parthood with some other relationship: based on the analysis for *ProChainS*, we now can make precise and motivate in Section 4.2 the intuition of what is called "transfers through" [15], "propagates_via" or "inheritance across transitive roles" [26] and "upward" and "right" "distributivity" over parthood relations [1] . Overall, it provides an indication for using *ProChainS* to *prevent* flaws.

4.1 Chaining typical properties in top-level ontologies

Many different object properties are declared in ontologies, but several re-occur across ontologies, such as on causality, agency, location, having a particular function or role, and inherence and having a certain quality. Such recurring types of relations typically are present also in foundational ontologies, such as DOLCE [21] and GFO [10], where they generally have domain and range axioms declared. In addition to considering the meaning of the relation, this feature makes them exceedingly suitable for evaluation with *ProChainS* even before declaring any property chain and property's usage in class expressions. Analysing the non-parthood relations, there are only a few safe property chains that can be declared at the time of writing, which is due to a combination of under- and over-specification.

Let us first consider a set of least constrained properties: the Relation Ontology (RO) [31]. RO is an ontology of relations and relationships that is used in bio-ontologies, especially within the OBO Foundry [30]. Based on those informal definitions, omitting is_a, inverse relations, subtypes of part_of, and instantiation, the meaningful property chains are indicated with a "+" in Table 4, "–" as definitely incorrect, and for the "±" cases it is unclear and they require either a more precise specification or user intervention on a case-by-case basis. Note that, with the present definitions [31] and its online version[7], part_of requires caution, because it uses the same name for continuant-parts, for process-parts, and for region-parts, which should not be mixed (recall Example 3), and the relations in the online formats do not have a user-defined domain and range[8]; e.g., a $\mathsf{part\_of} \circ \mathsf{has\_participant} \sqsubseteq \mathsf{part\_of}$ never can be

[7] `http://obofoundry.org/ro/`; last accessed on: 20-12-2012.

[8] This is being updated such that it is being integrated with BFO as the impending BFO v2, which diverges

correct due to the mismatch of $R_{\mathsf{part\_of}}$, a continuant, and $D_{\mathsf{has\_participant}}$, which is a process that is disjoint from continuant (recall also Evaluation 4). Overall, of the 81 options, only 9 are 'safe' to chain and return ontologically correct deductions, and 6 of them are so only thanks to their transitivity, therewith reducing the safe options to a mere 3. When more relations are defined and fully encoded in OWL, additional ontologically correct combinations may be identified, and a similar table can be constructed for the pattern $\mathsf{S} \circ \mathsf{R} \sqsubseteq \mathsf{R}$, which covers cases such as part_of ∘ participates_in ⊑ participates_in that are safe. Nevertheless, it is a low yield of possible chains.

Table 4 Reasoning with property chains for the relations in the Relation Ontology, with the $\mathsf{R} \circ \mathsf{S} \sqsubseteq \mathsf{R}$ pattern. Notes: when part_of is chained, then the domain and range are assumed to be the same on both sides of the inclusion, location (entailing spatial parthood) and containment (not entailing parthood) are taken here at the class-level.

$R \circ S \sqsubseteq R$ S right → R down ↓	part_of	located_in	contained_in	adjacent_to	transformation_of	derives_from	preceded_by	has_participant	has_agent
part_of	+	-	-	-	-	-	-	-	-
located_in	+	+	-	-	-	-	-	-	-
contained_in	+	+	+	-	-	-	-	-	-
adjacent_to	±	±	±	-	-	-	-	-	-
transf._of	-	-	-	-	+	-	-	-	-
derives_from	-	±	-	±	±	+	-	-	-
preceded_by	±	-	-	-	-	-	+	-	-
has_particip.	-	-	±	-	-	-	-	-	-
has_agent	-	-	-	-	-	-	-	-	-

The outcome of the analysis is even poorer for the more detailed formalisations of the relations in the foundational ontologies. We analysed from the `DOLCE-lite` OWL file the following object properties: constituent, participant, has-quality, q-location, r-location, physical-location-of, exact-location, and spatio-temporally-present-at and their inverses. From the GFO and the `gfo-basic` OWL file, we analysed the object properties: participates_in, plays_role, has_role, occupied_by, exhibits, and has_property and their inverses, where applicable. For GFO, one could add a chain that if a perpetuant exhibits some presential that occupies some space, then that perpetuant occupies that space, i.e., exhibits ∘ occupies ⊑ occupies; the user-defined domains and ranges for the other properties and incompatible. Regarding DOLCE, an interesting annotation in the `DOLCE-lite` file for r-location is "heterogeneous composition across physical and non-physical endurants", i.e.,

q-location-of ∘ inherent-in ∘ specific-constant-dependent ∘ has-quality ∘ q-location ⊑ r-location,

but the problem is that the annotation assumes that specific-constant-dependent has as domain a physical endurant (PED) and as range a non-physical endurant (NPED), whereas the relation itself is typed with only particular (PT), and inherent-in assumes a physical endurant as range in the chain, but its range axiom is asserted to be the top-level class particular. Put differently, on the one hand, the properties in these foundational ontologies are typed such that it severely limits the possibilities to chain, and on the other hand, they are not specific enough for the desired property chain to guarantee there will be no undesirable deductions.

Overall, this raises the observation that declaring useful property chains that do not involve parthood are more relevant for more precisely defined object properties in domain ontologies rather than broad-sweeping chains with more generic relations. Because properties cannot be anticipated, then the only recourse is, upon declaring a chain, to check it with *ProChainS*.

4.2 Property chaining with parthood

Some combinations of property chains are undesirable because it will lead to problematic deductions in any case, as with Mary and her legominifigure, or at least in some cases it is undesirable, such as with SNOMED CT's amputations and parthood, as introduced in Section 1, yet in other cases it is required, as with the femur's fracture and the drug treatments in the pharmacogenomics ontology. This does not mean, however, that it is all arbitrary and dependent on the situation. The main questions to answer, are how to 'remove' the seeming arbitrariness and how one can get the modeller to choose the ontologically correct options among alternatives, therewith to some extent pre-empting the need for *ProChainS*. We shall first address the recurring issue with parthood and containment, thereby providing a general scope—of which Mary and her mouth and minifigure are only one example—and then proceed to the more general case.

We assume the modeller understands part-whole relations, has typed the properties with domain and range axioms, has chosen the correct part-whole relationship between any two classes (e.g., with OntoPartS [19]), and is interested in chaining properties. Consider now

from the new relation ontology (`http://code.google.com/p/obo-relations/`; last accessed on: 20-12-2012) that is tailored to biology and has many object properties.

the four possible permutations with parthood and (3-dimensional) containment as a mereotopological relation between *regions with co-locating objects* (e.g., [19, 32]), depicted in the top-row of Figure 6. We can now rule out a chain like Figure 1-B. This becomes more evident with structural parthood between objects, i.e., structPartOf ∘ containedIn ⊑ structPartOf is wrong, whereas structPartOf ∘ containedIn ⊑ containedIn yields correct inferences. The reason for this is that the 'regions with co-locating objects' as domain and range for contains/containedIn—not representable in OWL but part of the relation's meaning nevertheless—does not enforce the constraint that one of the participating co-located objects must be a structural part of another (only a regional part), hence, ontologically by the intended meaning of the relations, one cannot infer that that must hold. The same argument holds when one substitutes containedIn for (2-dimensional) locatedIn.

The possibilities of chaining parthood with other relations is easier to assess when domain and range can be fully declared in an OWL ontology and observing whether the range of one property is compatible with the other(s). For instance, a parthood relation between processes (perdurants, PD), involvedIn ⊑ PD × PD [18], cannot chain in any arbitrary combination with participatesIn ⊑ ED × PD that relates an endurant to a perdurant; see Figure 6, third row.

The emerging pattern of safe and flawed chains is depicted in Figure 6. What this really means, is that out of the four permutations, at most two may be 'ontologically correct', or at least result in meaningful, intended, deductions. They are of the pattern

partOf ∘ [someOtherProperty] ⊑ [someOtherProperty]
and

[someOtherProperty] ∘ partOf ⊑ [someOtherProperty]
where [someOtherProperty] ⊑ ¬partOf and their respective domains/ranges are compatible. Phrased in natural language, the following *is likely to be correct* depending on the [someOtherProperty]:

- if the whole does/has [someOtherProperty], then we can infer that so does its part does/have [someOtherProperty], i.e., a 'downward distributivity' of the property of the whole to its part along the partonomy, and
- if some object does/have [someOtherProperty], then this distributes 'upward' so that the whole and upward in the partonomy also has the property that the part has, where the whole is delimited as such with a class (universal/concept) in the OWL ontology.

Conversely, it is *generally false* that parthood distributes either upward or downward over the [someOtherProperty]. Relating this back to Figure 1 where we have seen two options for chaining parthood and containment, Figure 1-A is an example with distributivity of [someOtherProperty] over a partonomy, whereas Figure 1-B is an example with distributivity of parthood over [someOtherProperty]. Thus, we have another argument in addition to domain/range axioms as to why Figure 1-A is better than Figure 1-B.

This insight allows a modeller to narrow down the options from four to at most two chains, where, depending on the need, both may be applicable, or only one of the two is the 'best' property chain. Here, the notion of 'best' refers to 'in any arbitrary situation', not a single, one-off in-my-ontology case. Take, for instance the car parts and ownership [13]: partOf ∘ owns ⊑ owns works with car and engine (if you own the car, you also own the engine that is part of the car) and possibly also the other way around, owns ∘ partOf ⊑ owns (if you own the car engine, it is fair to assume you also own the car), but the latter does not hold in all cases, for if one owns, say, a single apartment, one does not automatically also own the whole apartment building. Likewise, for the amputations and congenital absences in SNOMED CT, as mentioned in Section 1, point (iv): it is indeed a correct derivation that when the foot is amputated, so is the toe that was part of the foot, i.e. a downward distributivity—if you lose the whole, you lose its parts—but not an upward one. Thus, the upward and downward distributivity now also have a clear pattern of representation as to which property chains is applicable in the ontology. The underlying principle why in some cases only one of the two options for distributivity holds remains an open issue, and thus this last aspect has to be evaluated on a case-by-case basis for the time being.

5 Conclusions

We analysed and proposed the fundamental principles of subsumption of relationships in a relationship hierarchy, and integrated its results with OWL 2's basic and complex object property expressions. This enabled the identification of the type of flaws that can occur in the OWL object property box regarding simple property subsumption and property chaining, for which two compatibility services, *SubProS* and *ProChainS*, were proposed, which check for meaningful object property hierarchies and property chaining. Thanks to pinpointing the root cause with *SubProS* and *ProChainS*, 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. These insights were used to examine prevention of flaws and how to choose the best option among alternatives.

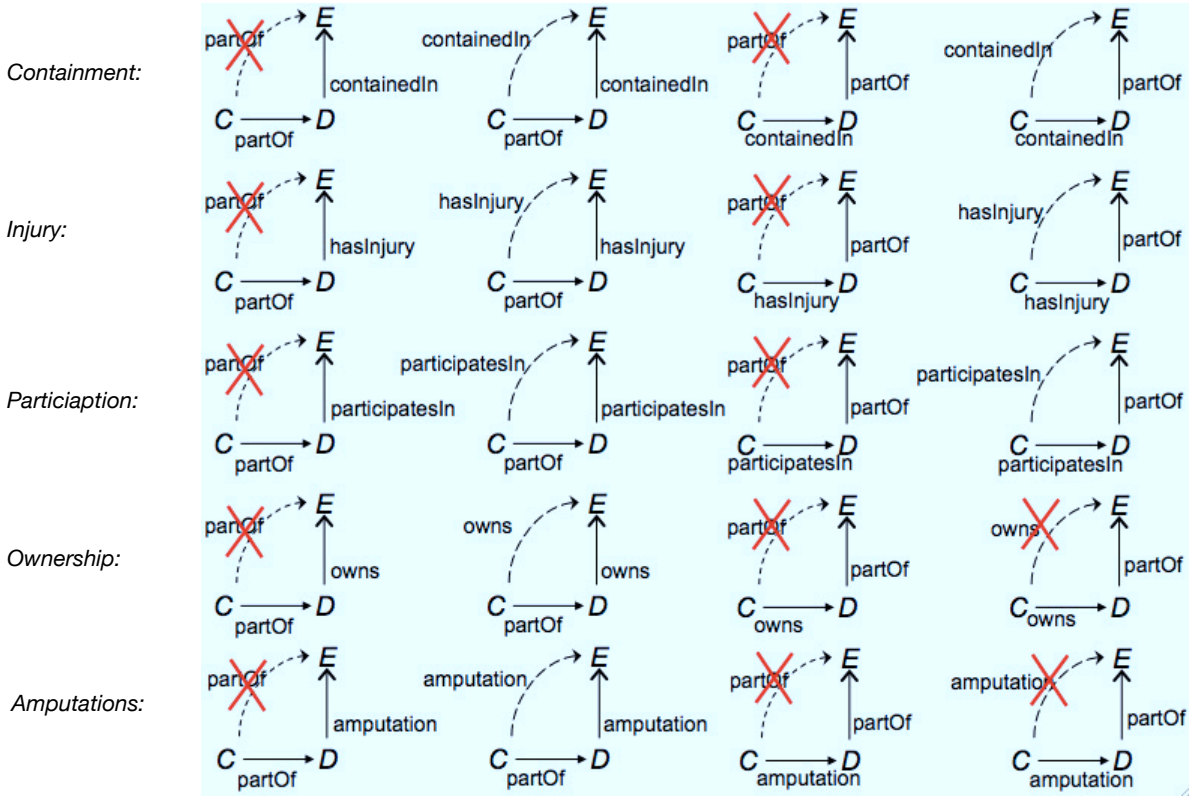

Fig. 6 Examples of simple property chaining with partOf and another property, its four options, and the remaining one or two that are ontologically correct, also demonstrating a pattern of distributivity (second column: downward distributivity; fourth column: upward distributivity).

Options for prevention are limited, but choosing among alternatives is made easy, especially with the chain pattern for upward and downward distributivity over parthood relations. *SubProS* and *ProChainS* were evaluated with several domain ontologies, which demonstrated that such flaws do exist, that they can be isolated effectively, and successful suggestions for revisions can be proposed.

We have commenced looking into an efficient algorithm to implement *SubProS* and *ProChainS* and a user-friendly interface to help revising flaws for the ontology development environment. Another avenue can be to extend OntoPartS [19] for advanced modelling of part-whole relations—which is currently redesigned as a Protégé plugin—with *ProChainS*.

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

References

1. Artale, A., Franconi, E., Guarino, N., Pazzi, L.: Part-whole relations in object-centered systems: An overview. Data and Knowledge Engineering **20**(3), 347–383 (1996)
2. 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), 205–212 (2008)
3. 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
4. Cuenca Grau, B., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P., Sattler, U.: OWL 2: The next step for OWL. Journal of Web Semantics: Science, Services and Agents on the World Wide Web **6**(4), 309–322 (2008)
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. Glimm, B., Rudolph, S., Völker, J.: Integrated metamodeling and diagnosis in OWL 2. In: P.F. Patel-Schneider, Y. Pan, P. Hitzler, P. Mika, L. Zhang, J.Z. Pan, I. Horrocks, B. Glimm (eds.) Proceedings of the 9th International Semantic Web Conference, *LNCS*, vol. 6496, pp. 257–272. Springer (2010)
7. Guarino, N., Welty, C.: An overview of OntoClean. In: S. Staab, R. Studer (eds.) Handbook on ontologies, pp. 151–159. Springer Verlag (2004)
8. Halpin, T.: Information Modeling and Relational Databases. San Francisco: Morgan Kaufmann Publish-

ers (2001)
9. Halpin, T.A., Curland, M.: Enriched support for ring constraints. In: R. Meersman, T.S. Dillon, P. Herrero (eds.) OTM Workshops 2011, *LNCS*, vol. 7046, pp. 309–318. Springer (2011). Hersonissos, Crete, Greece, October 17-21, 2011
10. Herre, H.: General Formal Ontology (GFO): A foundational ontology for conceptual modelling. In: R. Poli, M. Healy, A. Kameas (eds.) Theory and Applications of Ontology: Computer Applications, chap. 14, pp. 297–345. Springer, Heidelberg (2010). DOI 10.1007/978-90-481-8847-5_14
11. Hilario, M., Nguyen, P., Do, H., Woznica, A., Kalous, A.: Ontology-based meta-mining of knowledge discovery workflows. In: N. Jankowski, W. Duch, K. Grabczewski (eds.) Meta-learning in Computational Intelligence, pp. 273–315. Springer (2011)
12. Horridge, M., Parsia, B., Sattler, U.: Laconic and precise justifications in OWL. In: Proc. of the 7th International Semantic Web Conference (ISWC 2008), *LNCS*, vol. 5318. Springer (2008)
13. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible $\mathcal{SROIQ}$. Proceedings of KR-2006 pp. 452–457 (2006)
14. Horrocks, I., Patel-Schneider, P.F., van Harmelen, F.: From SHIQ and RDF to OWL: The making of a web ontology language. Journal of Web Semantics **1**(1), 7 (2003)
15. Horrocks, I., Rector, A., Goble, C.: A description logic based schema for the classification of medical data. In: Proc. of the 3rd Int. Workshop on Knowledge Representation meets Databases (KRDB'96), *CEUR-WS*, vol. 4, pp. 24–28 (1996)
16. Kalyanpur, A., Parsia, B., Sirin, E., Grau, B.: Repairing unsatisfiable concepts in OWL ontologies. In: Y. Sure, J. Domingue (eds.) Proceedings of the European Semantic Web Conference (ESWC'06), *LNCS*, vol. 4011. Springer (2006)
17. Keet, C.M.: Detecting and revising flaws in OWL object property expressions. In: A. ten Teije, et al. (eds.) 18th International Conference on Knowledge Engineering and Knowledge Management (EKAW'12), *LNAI*, vol. 7603, pp. 252–266. Springer (2012). Oct 8-12, Galway, Ireland
18. Keet, C.M., Artale, A.: Representing and reasoning over a taxonomy of part-whole relations. Applied Ontology **3**(1-2), 91–110 (2008)
19. Keet, C.M., Fernández-Reyes, F.C., Morales-González, A.: Representing mereotopological relations in OWL ontologies with ONTOPARTS. In: E. Simperl, et al. (eds.) Proceedings of the 9th Extended Semantic Web Conference (ESWC'12), *LNCS*, vol. 7295, pp. 240–254. Springer (2012). 29-31 May 2012, Heraklion, Crete, Greece
20. 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). URL `http://journals.tdl.org/jodi/article/viewArticle/693/577`
21. Masolo, C., Borgo, S., Gangemi, A., Guarino, N., Oltramari, A.: Ontology library. WonderWeb Deliverable D18 (ver. 1.0, 31-12-2003). (2003). Http://wonderweb.semanticweb.org
22. 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), pp. 193–198 (2001)
23. Motik, B., Patel-Schneider, P.F., Parsia, B.: OWL 2 web ontology language structural specification and functional-style syntax. W3c recommendation, W3C (2009). Http://www.w3.org/TR/owl2-syntax/
24. 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
25. Poveda-Villalón, M., Suárez-Figueroa, M.C., Gómez-Pérez, A.: Validating ontologies with OOPS! In: A. ten Teije, et al. (eds.) 18th International Conference on Knowledge Engineering and Knowledge Management (EKAW'12), *LNAI*, vol. 7603, pp. 267–281. Springer (2012). Oct 8-12, Galway, Ireland
26. Rector, A.: Analysis of propagation along transitive roles: Formalisation of the GALEN experience with medical ontologies. In: I. Horrocks, S. Tessaris (eds.) Proceedings of the International Workshop on Description Logics (DL'02), *CEUR-WS*, vol. 53 (2002). Toulouse, France, April 19-21, 200
27. Rector, A., Drummond, N., Horridge, 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), *LNCS*, vol. 3257, pp. 63–81. Springer (2004). Whittlebury Hall, UK
28. Roussey, C., Corcho, O., Vilches-Blázquez, L.: A catalogue of OWL ontology antipatterns. In: Proc. of K-CAP'09, pp. 205–206 (2009)
29. Schmidt-Schauss, M.: Subsumption in KL-ONE is undecidable. In: Proceedings of 1st Conference on Knowledge Representation and Reasoning (KR'89), pp. 421–431 (1989)
30. Smith, B., Ashburner, M., Rosse, C., Bard, J., Bug, W., Ceusters, W., Goldberg, L., Eilbeck, K., Ireland, A., Mungall, C., OBI Consortium, T., Leontis, N., Rocca-Serra, A., Ruttenberg, A., Sansone, S.A., Shah, M., Whetzel, P., Lewis, S.: The OBO Foundry: Coordinated evolution of ontologies to support biomedical data integration. Nature Biotechnology **25**(11), 1251–1255 (2007)
31. Smith, B., Ceusters, W., Klagges, B., Köhler, J., Kumar, A., Lomax, J., Mungall, C., Neuhaus, F., Rector, A.L., Rosse, C.: Relations in biomedical ontologies. Genome Biology **6**, R46 (2005)
32. Varzi, A.: Handbook of Spatial Logics, chap. Spatial reasoning and ontology: parts, wholes, and locations, pp. 945–1038. Berlin Heidelberg: Springer Verlag (2007)
33. Wessel, M.: Obstacles on the way to qualitative spatial reasoning with description logics: some undecidability results. In: C.A. Goble, D.L. McGuinness, R. Möller, P.F. Patel-Schneider (eds.) Proceedings of the International Workshop in Description Logics (DL'01), *CEUR WS*, vol. 49 (2001). Stanford, CA, USA, August 1-3, 2001