

Representing Transitive Propagation in OWL

Julian Seidenberg and Alan Rector

Medical Informatics Group
University of Manchester
United Kingdom

`jms@cs.manchester.ac.uk`, `rector@cs.manchester.ac.uk`

Abstract. Transitive propagation along properties can be modelled in various ways in the OWL description logic. Doing so allows existing description logic reasoners based on the tableaux algorithm to make inferences based on such transitive constructs. This is especially useful for medical knowledge bases, where such constructs are common.

This paper compares, contrasts and evaluates a variety of different methods for simulating transitive propagation: property subsumption, classic SEP triples and adapted SEP triples. These modelling techniques remove the need to extending the OWL language with additional operators in order to express the transitive propagation. Other approaches require an extended tableaux reasoner or first-order logic prover, as well as a modification of the OWL standard.

The adapted SEP triples methodology is ultimately recommended as the most reliable modelling technique.

1 Introduction

1.1 Transitivity

A transitive relation is a relation between three elements if it holds between the first and second and it also holds between the second and third it must necessarily hold between the first and third [1].

Transitivity is one of the three intrinsic properties of part/whole relations. Winston calls this “a single sense of part” [2]: if the door is part of the car and the door-handle is part of the door, then the door-handle is also part of the car. If (A *isPartOf* B) and (B *isPartOf* C) then (A *isPartOf* C).

1.2 Transitive propagation

However, the above does not necessarily hold true universally. Odell [3] points out that there are many different kinds of composition. When we say “part of” we often mean very different things. For example, “Iron isPartOf Car” implies a material-object relation, i.e. the car object is made of the iron material, while “Car isPartOf Traffic” implies a member-bunch relation, i.e. the car is a member of the collection of things which make up the Traffic concept.

$(Piston\ isPartOf\ Engine) \sqcap (Engine\ isPartOf\ Car) \rightarrow (Piston\ isPartOf\ Car)$

Fig. 1. Transitive relation because of similar semantics

$(Piston\ isPartOf\ Car) \sqcap (Car\ isPartOf\ Traffic) \not\rightarrow (Piston\ isPartOf\ Traffic)$

Fig. 2. Non-transitive relation because of different semantics

While each specific type of part/whole relation is transitive along relations with the same semantics, as illustrated in Figure 1, this does not necessarily hold true across different types of relations, as shown in Figure 2.

However, as will be explained in the next section, in some cases, *transitive propagation* (sometimes also called a *role path*, or *propagates-via*) along relations with different semantics is desirable.

(Note: up to this point we referred to all relations as “partOf” in order to illustrate transitivity. However, for the purpose of more clearly distinguishing between relations with different semantics, we will proceed to name them more descriptively.)

1.3 Related work

$Burn \sqsubseteq \exists\ isLocatedIn.\ Toe$
 $Toe \sqsubseteq \exists\ isStructuralComponentOf.\ Foot$

Fig. 3. Motivating example for transitive propagation in OWL

In the example in Figure 3, one would intuitively expect a knowledge base to know that the Burn is located in the Foot, as well as the Toe (since the latter is a part of the former). However, Horrocks and Patel-Schneider point out that the description logic (DL) based Web Ontology Language (OWL) [4] currently does not support this kind of inference. Their proposed solution is to extend OWL with a rules language to, among other things, model transitive propagation [5]. Automated reasoning in such an extended OWL language requires a first-order logic or hybrid prover system. Existing tableaux algorithm based reasoners do not suffice. However, reasoning using such a system is provably undecidable [6].

Another solution, adopted by SNOMED [7] and GALEN [8] medical terminologies, is to introduce an idiom for transitive propagation into the modelling formalism. GALEN, for example, uses two special operators (“specialisedBy” and “refinedAlong”) to express transitive propagation between roles [9].

In fact, the draft proposal for OWL 1.1 [10] proposes to extend the expressive power of OWL DL from *SHOIN* to the *SRQIQ* description logic [11], which, among other things, allows for transitive propagation using, so called, *property chain inclusion axioms*.

However, it may be some time until the OWL 1.1 standard is ratified and implemented [12]. If and when this occurs it will be interesting to compare the reasoning performance of that implementation of transitive propagation to that of the various models which are currently available. The methods described in this paper will then, at worst, be useful for purposes of backwards compatibility and, at best, offer better performing, though less convenient, solution.

The following sections explain, compare, contrast and evaluate several ways of modelling transitive propagation, all of which avoid the need to extend the current OWL 1.0 language and are decidable using current reasoning tools.

2 Styles of transitive propagation

A specific illustrative example will be used throughout this paper to show the different styles of modelling transitive propagation in OWL. Assuming the knowledge base in Figure 4 is given:

```

Foot  $\sqsubseteq \exists$  isPartOf. Leg
Toe  $\sqsubseteq \exists$  isPartOf. Foot
Burn  $\sqsubseteq \exists$  isLocatedIn. Toe
LegInjury  $\equiv \exists$  isLocatedIn. Leg

```

Fig. 4. Initial example ontology

The defined class in the last line of Figure 4 serves as a query. It should subsume all possible injuries to the *Leg* when the knowledge base is classified. That is, once all implicit relationships in the ontology are made explicit, and assuming transitive propagation is properly modeled, then *Burn* should be subsumed under *LegInjury*.

2.1 Property subsumption

One way of simulating transitive propagation is to use the property hierarchy to assert one property as a subproperty of another. For example, if “*isLocatedIn* propagatesVia *isPartOf*”, then “*isPartOf* subsumes *isLocatedIn*”, where both *isLocatedIn* and *isPartOf* are transitive properties. This is shown more formally in Figure 5.

$$r \circ s \stackrel{\dot{\sqsubseteq}}{\sqsubseteq} r \Rightarrow s \sqsubseteq r \quad (\circ \text{ indicates transitive propagation})$$

$$s \in R_+ \quad (R_+ \text{ is the set of transitive property names})$$

Fig. 5. Simulating transitive propagation by using the property hierarchy

This method is easy to understand, simple to implement and, as will be shown later in section 3, provides good performance. However, it also has numerous disadvantages.

As pointed out by Rector in [13], a tangled ontology is very difficult to maintain. Tangled ontologies have subsumption hierarchies with more than one superclass per class. The maintenance difficulty is equally applicable to a hierarchy of properties. Using property subsumption to simulate transitive propagation in ontologies with large numbers of properties can therefore quickly lead to an unmaintainable knowledge base. In such cases, the information about mutual propagation among properties is best kept externally and applied to ontology using a script. JOT [14], for example, is well suited for this purpose.

Another disadvantage of this method is that its logical meaning is inaccurate. Unexpected logical inferences are therefore sometimes possible. For example, given the knowledge bases in Figure 4 and the property subsumption method, as outlined above ($isLocatedIn \circ isPartOf \sqsubseteq isLocatedIn \Rightarrow isPartOf \sqsubseteq isLocatedIn$), the query for *LegInjury* would result in both *Burn* and *Toe*. That is, both concepts would be inferred as subclasses of *LegInjury*, since any *isPartOf* relation is also an *isLocatedIn* relation.

Inferring *Toe* as a subclass of *LegInjury* is obviously not the intended meaning, but may be acceptable in some cases. For example, further restricting the query by adding more information, as shown in Figure 6, yields the expected result. The property subsumption technique for transitive propagation certainly requires careful analysis and should never be applied blindly.

$$\begin{aligned} Burn &\sqsubseteq Injury \\ Burn &\sqsubseteq \exists isLocatedIn.Toe \\ LegInjury &\equiv \left(\begin{array}{l} Injury \sqcap \\ \exists isLocatedIn.Leg \end{array} \right) \end{aligned}$$

Fig. 6. Correctly behaving *LegInjury* query using property subsumption

2.2 Classic SEP triples

Schulz and Hahn introduce the idea of SEP triples [15]. Their idea allows transitivity to be modeled explicitly. That is, SEP triples enable transitive relations to be expressed in formalisms that do not include transitivity by explicitly distinguishing between the whole of a concept, parts of a concept and the disjunction of the whole of a concept and its parts. Details of these triples may be found in [15].

An implementation of classic SEP triples requires extensive modification of the ontology class hierarchy. Three separate classes need to be introduced for every actual concept in the knowledge base. This results in a complex ontology structure that is difficult to maintain. Furthermore, we do not know of any algorithm for creating an ontology with SEP triples from a base ontology, given

a list of transitively propagating properties. The performance of classic SEP triples was therefore not evaluated as part of this research. However, we presume their performance is inferior to that of the adapted SEP triples methodology (see below), since they do not take full advantage of OWL.

2.3 Adapted SEP triples

Rector suggests an adapted SEP triples formalism [9] for use in description logics with transitive properties such as OWL ($\mathcal{SHOIN}(\mathcal{D})$) [16].

Similar to classic SEP triples, this methodology explicitly models transitive propagation in the knowledge base. However, unlike Schulz and Hahn’s original idea, adapted SEP triples take advantage of OWL’s ability to represent transitive properties. This removes the need to model transitivity explicitly in the knowledge base and therefore allows a much cleaner SEP triple-like representation to be created.

$$\begin{aligned}
 isLocatedIn &\in R && (R \text{ is the set of all property names}) \\
 isPartOf &\in R_+ && (R_+ \text{ is the set of transitive properties}) \\
 R_+ &\subseteq R \\
 LegInjury &\equiv \exists isLocatedIn. \left(\begin{array}{c} Leg \sqcup \\ \exists isPartOf. Leg \end{array} \right)
 \end{aligned}$$

Fig. 7. Adapted SEP triples query

Example Reusing the example knowledge base from Figure 4 above and classifying it together with the the query in Figure 7, results in the the expected inference: *Burn* is found to be a subclass of *LegInjury*.

Figure 8 shows how this adapted SEP triples mechanism works:

Toe, *Foot* and *Leg* are all concepts that are transitively part of each other, as indicated by the solid, upwards arcing arrows. There is also the *Burn* concept located in the *Toe*. Additionally, the model contains *LegInjury*, which is the defined class from Figure 7 that captures all things located in the *Leg*, or any of its parts.

Since *isPartOf* is a transitive property, the *Toe* concept is also part of the *Leg* concept. Therefore, anything located in the *Toe* (such as the *Burn*) matches the second part of the definition of *LegInjury*. That is, it is located in something which is part of the *Leg*. This results in *Burn* being inferred as a subclass of *LegInjury* when the knowledge base is classified by a description logic reasoner, as indicated by the dotted, straight, upwards pointing arrow.

Constraints and assumptions All knowledge bases used as examples in this paper are assumed to be normalised [13]. Defined classes act as queries in ontologies built using these principles. New subsumption relations are inferred only for

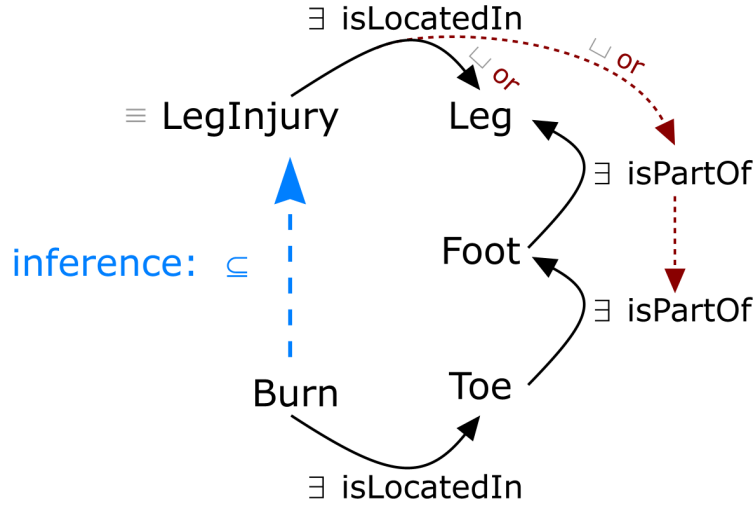


Fig. 8. Adapted SEP triples in action

such classes. Therefore only the defined classes in such an ontology need to be modified in order to create adapted SEP triples. However, in arbitrary ontologies SEP triples need to be applied to all classes in order to achieve a logically complete solution.

Transformations Figure 9 shows the transformations that need to be applied to the defined classes in a knowledge base in order to create adapted SEP triples.

$$\begin{aligned}
 & \text{for: } R \circ S \stackrel{\subseteq}{\subseteq} R \quad (\text{where } S \text{ is a transitive property}) \\
 \exists R.C & \Rightarrow \exists R.(C \sqcup \exists S.C) \\
 \forall R.C & \Rightarrow \forall R.(C \sqcup \exists S.C) \\
 \exists R^-.C & \Rightarrow (\exists R^-.C) \sqcup (\exists S^-.(\exists R^-.C)) \\
 \forall R^-.C & \Rightarrow \frac{\forall R^-.C}{\exists S^-.T \sqsubseteq \forall R^-.C}
 \end{aligned}$$

Fig. 9. Transformations for creating adapted SEP triples

The last transformation rule requires some explanation: a class transformed in this way captures all classes that match the basic inverse restriction, while also being restricted to some other class in the ontology (T) via the secondary property (S), where that other class must also have the same basic restriction as its superclass.

Handling multiple transitive propagations Chains of defined classes require special consideration as the rules in Figure 9 must be applied recursively. That is: when one defined class references another defined class, the transformed SEP triple restriction no longer matches the original definition. The original defined class must therefore be transformed to match the newly transformed definition of the second defined class. Chains of defined classes do not classify correctly without this additional transformation.

$$\begin{aligned}
 & isLocatedIn \in R \\
 & isPartOf \in R_+ \\
 & isMultipleOf \in R_+ \\
 & LegInjury \equiv \exists isLocatedIn. \left(\exists isPartOf. \left(\begin{array}{l} Leg \sqcup \\ Leg \sqcup \\ \exists isMultipleOf. Leg \end{array} \right) \right)
 \end{aligned}$$

Fig. 10. New query for adapted SEP triples with chained definitions

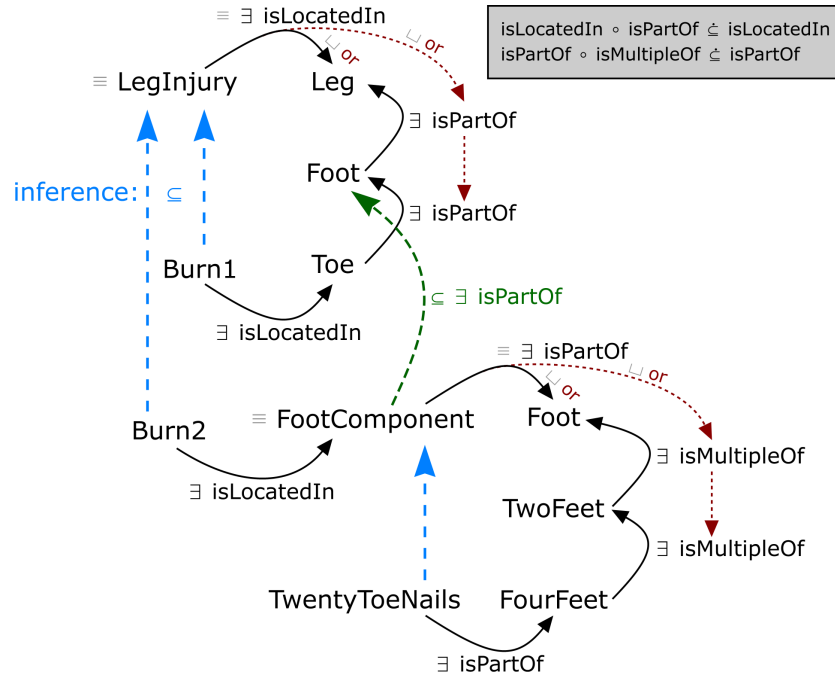


Fig. 11. Example of multiple transitive propagations

Figure 11 gives an example of such a case: suppose we take the ontology from Figure 8 and add a second SEP triple definition. If only the necessary and suffi-

cient condition on the *FootComponent* class ($\equiv \exists isPartOf.(Foot \sqcup \exists isMultipleOf .Foot)$) is asserted, then the link from the *FootComponent* to the original *Foot* concept does not hold and the SEP triple inference cannot take effect; i.e. the *Burn2* concept is not classified correctly. However, if an additional transformation is applied to *LegInjury*, resulting in the new definition of that class as shown in Figure 10, then the link indicated by the striped curved upwards pointing arrow is captured and the correct inference results. That is: *Burn2* is inferred as being a kind of *LegInjury*.

It should be noted that these kinds of chained universal restrictions may not need to be taken into account when creating SEP triples, depending on the ontology in question. However, some medical ontologies (such as GALEN) contain a substantial amounts of transitive propagation. A correct implementation is crucial in these cases.

Rector neglects to mention the need for a recursive algorithm when originally describing adapted SEP triples [9].

Discussion Advantages of this modelling methodology are that it is logically correct and therefore, unlike the property subsumption method, will not produce any unexpected behaviour. It can also be applied selectively (unlike the potential implementation in the *SRONTQ* description logic [11] based on complex property chain inclusion axioms [12]), so some concepts in an ontology can use transitive propagation, while others do not.

However, adapted SEP triples (unlike property subsumption) modify concept semantic (though not as drastically as classic SEP triples do) and may therefore be more difficult for a beginner to comprehend. They also require a somewhat complicated recursive transformation algorithm when dealing with chained transitive propagation.

3 Evaluation

Description logic reasoners such as FaCT++ [17], RACER [18], or Pellet [19] can be used to infer information that is implicit in an ontology [20].

3.1 Test setup

Classification speed tests were carried out using the RACER 1.8 description logic reasoning system [18] on a 2.8 Ghz Pentium 4 with 2.5 GB of RAM running Windows XP. All tests were carried out utilizing the maximum memory possible in 32-bit Java applications (1.5 GB). The figures quoted are the times spent in actual reasoning. Data transfer latency is not shown. Tests were run for as long as necessary. That is, classification failure is reported only if the reasoner application crashed while attempting to classify a particular ontology.

3.2 Ontology segment test sets

None of the description logic reasoners based on the tableaux algorithm mentioned above are currently able to classify the complete GALEN ontology. GALEN is too large and complex for these reasoning systems. (Note: the original classifier used for GALEN was based on different principles and did not suffer from this particular limitation [21].)

The ontology segmentation algorithm described in [22] was therefore used to create a test set of smaller, classifiable segments of the complete GALEN ontology. This test set consisted of a total of 162 ontology extracts centred around the *Heart* concept from the GALEN ontology [8]. Segments were chosen so all the base-case extracts were tractable.

The GALEN ontology employs a rich property hierarchy with over 500 distinct property types. The top-level of this hierarchy forms a meta-property structure. Using this high-level grouping it was possible to selectively include and/or exclude different ranges of properties in various segments. The following individual meta-properties and their combinations were selected for evaluation:

- **modifierAttribute**: properties which can be used to modify a given class such as “color” or “status”. These are sometimes also known as “value partitions” [23]. They are not likely to adversely effect tractability, since they themselves do not contain further definitions.
- **locativeAttribute**: properties that link diseases to anatomical locations that they are in some way related to.
- **structuralAttribute**: properties linking anatomical body structures together by physical composition.
- **partitiveAttribute**: properties that link classes based on processes, divisions and other partitive relations
- **functionalAttribute**: properties that link classes by action or function.

(Note: a more detailed analysis of the GALEN property hierarchy may be found in [24].)

The GALEN ontology was filtered using four individual meta-properties (locative, structural, partitive, functional) as well as four combinations of meta-properties (functional + modifier, structural + modifier, partitive + functional + locative, structural + functional). These property sets were used to generate a various ontology segments. Additionally, the depth of the link traversal algorithm was limited in order to produce even more tightly constrained versions of these ontologies. Segments were created with maximal depths for the recursive ontology segmentation algorithm ranging from one to five, as well as without any depth limit. Finally, different styles of transitive propagation, based upon rules harvested from the original GALEN ontology, were applied to each extract (no transitive propagation, property subsumption and adapted SEP triples). This lead to a total of 144 test ontologies $((4 + 4) \times 6 \times 3 = 144)$.

3.3 Test evaluation

The following observations can be made from the tests shown in Figure 12:

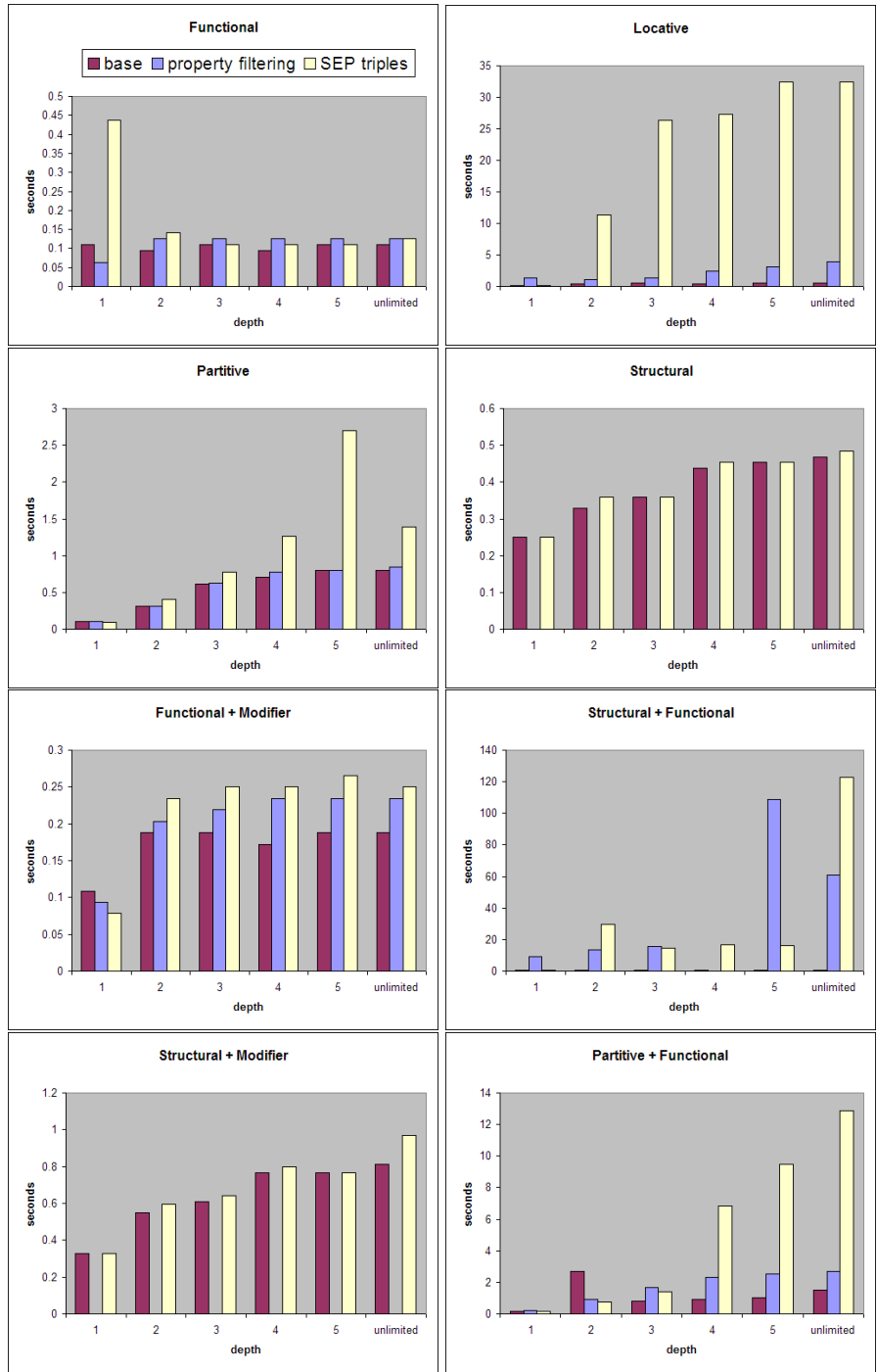


Fig. 12. Timing test results

- Classification performance for *functional* properties is similar regardless of the method used. This is due to the relatively small amount of transitive propagation in those extracts.
- Extracts transformed to employ SEP triples using *locative* properties take an order of magnitude longer to classify than those using *partitive* segments. This is in spite of the *partitive* extracts having more actual SEP triples (388 vs. 244 triples in the case of unlimited extract depth). One can therefore conclude that classification speed is not directly correlated with the number of triples, but is more complex of an issue. Indeed, in both cases, the property subsumption technique performs very well.
- All the *structural* segments that employ property subsumption transformations crash the reasoner.
- The slowest classification performance in the test set results from an extract combining the *structural* and *functional* properties. Both of these property sets can be classified individually within about a second. However, the combination performs up to one hundred times slower. A similar pattern can be observed from the combination of *partitive* and *functional* properties.
- Extracts filtered using *structural* properties are unclassifiable when using the property subsumption technique. However when *functional* properties and *structural* properties are combined, this combination extract suddenly becomes tractable. However, classification performance suffers by almost three orders of magnitude compared to *functional* properties on their own.

In summary: *structural* properties scale extremely badly for property subsumption. *Locative* properties scale badly for SEP triples. SEP triples performance is slower than property subsumption and far slower than classification without transitive propagation. In rare cases, adding more information/complexity causes previously intractable knowledge bases to become classifiable.

4 Discussion and Future Work

In this paper two techniques for modelling transitive propagation in the OWL description logic have been described. Both allow an ontology engineer to express simple rule-like constructs while avoiding the use of complex first-order logic rules languages.

The property subsumption method of adding transitive propagation to an ontology results in a tangled property hierarchy, which can create complex cycles. These cycles can make classification completely intractable. This technique is also logically inaccurate and can result in incorrect/unintended inferences. However, performance is only slightly worse than the base-case.

Adapted SEP triples, on the other hand, add a large number of disjunctions in the knowledge base, thereby increasing the number of possibilities a tableaux reasoning system must explore in order to classify the ontology [25]. This can result in a significant increase in classification time. However, unlike

the property subsumption mechanism, the increase in complexity does not result in intractability and is logically correct.

Adapted SEP triples are more difficult to implement and slower to classify, but we nevertheless recommend their use, as they are a much safer and more predictable modelling technique.

Future work includes evaluating the performance of a native tableaux algorithm implementation of transitive propagation. This research will be carried out if and when the *SR_{CIQ}* description logic [11] (which is due to underlie OWL 1.1) with its complex role inclusion axioms is implemented in a reasoning system. When this occurs, backwards compatibility with legacy tools and applications can be preserved, by using the methods outlined in this paper.

References

1. TheFreeDictionary.com: Transitivity definition (2004)
2. Winston, M., Chaffin, R., Herrmann, D.: A taxonomy of part-whole relations. In: Cognitive Science. Volume 11. (1987) 417–444
3. Odell, J.J.: Six different kinds of composition. Journal of Object-Oriented Programming **5**(8) (1994) 10–15
4. Smith, M.K., Welty, C., McGuinness, D.L.: OWL web ontology language guide. W3C Recommendation (10 February 2004)
5. Horrocks, I., Patel-Schneider, P.F.: A proposal for an OWL rules language. In: Proc. of the Thirteenth International World Wide Web Conference (WWW 2004), ASCM (2004) 723–731
6. Horrocks, I., Patel-Schneider, P.F., Bechhofer, S., Tsarkov, D.: OWL rules: A proposal and prototype implementation. Journal of Web Semantics **3**(1) (2005) 23–40
7. Spackman, K.: Managing clinical terminology hierarchies using algorithmic calculation of subsumption: Experience with SNOMED-RT. Journal of the American Medical Informatics Association **Fall Symposium** (2000)
8. Rector, A.L., Bechhofer, S., Goble, C., Horrocks, I., Nowlan, W.A., Solomon, W.D.: The GRAIL concept modelling language for medical terminology. Artificial Intelligence in Medicine **9**(2) (1997) 139–171
9. Rector, A.: Analysis of propagation along transitive roles: Formalisation of the GALEN experience with medical ontologies. In: DL 2002. (2002)
10. Patel-Schneider, P.: The OWL 1.1 Extension to the W3C OWL Web Ontology Language (2005)
11. Horrocks, I., Kutz, O., Sattler, U.: The Even More Irresistible SROIQ. Technical report, University of Manchester (2005)
12. Horrocks, I., Sattler, U.: Decidability of *SHIQ* with complex role inclusion axioms. Artificial Intelligence **160**(1–2) (2004) 79–104
13. Rector, A.L.: Normalisation of ontology implementations: Towards modularity, re-use, and maintainability. In: EKAW Workshop on Ontologies for Multiagent Systems. (2002)
14. Dameron, O.: JOT: a Scripting Environment for Creating and Managing Ontologies. In: 7th International Protégé Conference. (2004)
15. Schulz, S., Hahn, U., Romacher, M.: Part-Whole Reasoning in Medical Ontologies Revisited: Introducing SEP Triplets into Classification-Based Description Logics. In: AMIA Annual Fall Symposium, Hanley & Belfus (1998) 830–834

16. Horrocks, I., Patel-Schneider, P.F., van Harmelen, F.: From SHIQ and RDF to OWL: The making of a web ontology language. In: *Journal of Web Semantics*. Volume 1. (2003) 7–26
17. Tsarkov, D., Horrocks, I.: Reasoner prototype: Implementing new reasoner with datatypes support. WonderWeb Project Deliverable (2003)
18. Haarslev, V., Möller, R.: RACER System Description. In Gor, R., Leitsch, A., Nipkow, T., eds.: *Automated Reasoning: First International Joint Conference*. Volume 2083 / 2001., Springer-Verlag Heidelberg (2001) 701
19. Parsia, B., Sirin, E.: Pellet: An OWL DL reasoner. ISWC 2004 (2004) ISWC.
20. Lutz, C., Sattler, U., Tendera, L.: The complexity of finite model reasoning in description logics. In: *Automated Deduction*, Springer-Verlag (2003) 60 – 74
21. Horrocks, I., Rector, A.L., Goble, C.A.: A Description Logic Based Schema for the Classification of Medical Data. In: *KRDB*. (1996)
22. Seidenberg, J., Rector, A.: Web ontology segmentation: Analysis, classification and use. In: *15th International World Wide Web Conference*. (2006)
23. Drummond, N., Horridge, M., Wang, H., Rogers, J., Knublauch, H., Stevens, R., Wroe, C., Rector, A.: Designing User Interfaces to Minimise Common Errors in Ontology Development: the CO-ODE and HyOntUse Projects. In Cox, S.J., ed.: *Proceedings of the UK e-Science All Hands Meeting*. (2004)
24. Rogers, J., Rector, A.: GALEN’s model of parts and wholes: Experience and comparisons. *Proceedings of AMIA Symposium* (2000) 714–8
25. Horrocks, I.: Optimisation techniques for expressive description logics. Technical Report UMCS-97-2-1, University of Manchester, Department of Computer Science (1997)