A Solution for the Man-Man Problem in the Family History Knowledge Base

Dmitry Tsarkov and Ulrike Sattler and Margaret Stevens and Robert Stevens

The University of Manchester, Manchester, UK
{tsarkov|sattler|stevens}@cs.man.ac.uk

Abstract. The Family History Knowledge Base (FHKB) was presented at OWLED in 2008. The FHKB uses a rich object property hierarchy, including many OWL 2 features, to derive many entailments on some 450 individuals representing the Stevens family. In the ABox, only parent relationships and some sibling relationships (either `isSisterOf` or `isBrotherOf`) are asserted. Using a sparse assertion of brother or sister relationships, together with information about gender, other sibling relationships should be able to be inferred. The inability to do this in OWL has been described as the 'Man-Man' problem, and various workarounds have been discussed. We describe a new solution to this issue, implemented in the reasoner FaCT++. This solution allows to capture axioms such as 'My male siblings are my brothers', and we have added them to the FHKB. The number of entailments about, for instance, sibling relationships increases significantly without increasing the number of asserted facts about members of the Stevens family.

1 Introduction

In this paper we discuss our latest insights gained while using OWL 2 to model a Family History Knowledge Base (FHKB) [8]. We use FHKB to illustrate a new extension and the inferences that can be drawn in the FHKB example.

The FHKB uses automated reasoning over facts asserted about relationships in a family to form a genealogy. Family history is an attractive example as it applies to each and every person and all family relationships can be determined from only information about parentage. The addition of information about marriage means that both blood- and relationships by marriage can be drawn. A rich property hierarchy, property characteristics and sub-property chains is used in order to maximise the number of entailments on an individual from the fewest possible assertions of facts on those individuals. The class and property hierarchies can be seen in Figure 1.

The FHKB has some 450 individuals describing the Stevens family history. Only assertions are made about motherhood and fatherhood, together with sparse assertion on brother- and sisterhood. Other relations can be entailed from these ones by the reasoner using the given property axioms. For example, by transitivity of the `isBrotherOf` property it is possible to ensure more siblings of a given person as having brothers. More complex relations (like `isUncleOf`)

Proceedings of OWL: Experiences and Directions 2009 (OWLED 2009),
Rinke Hoekstra and Peter F. Patel-Schneider, editors. http://www.webont.org/owled/2009

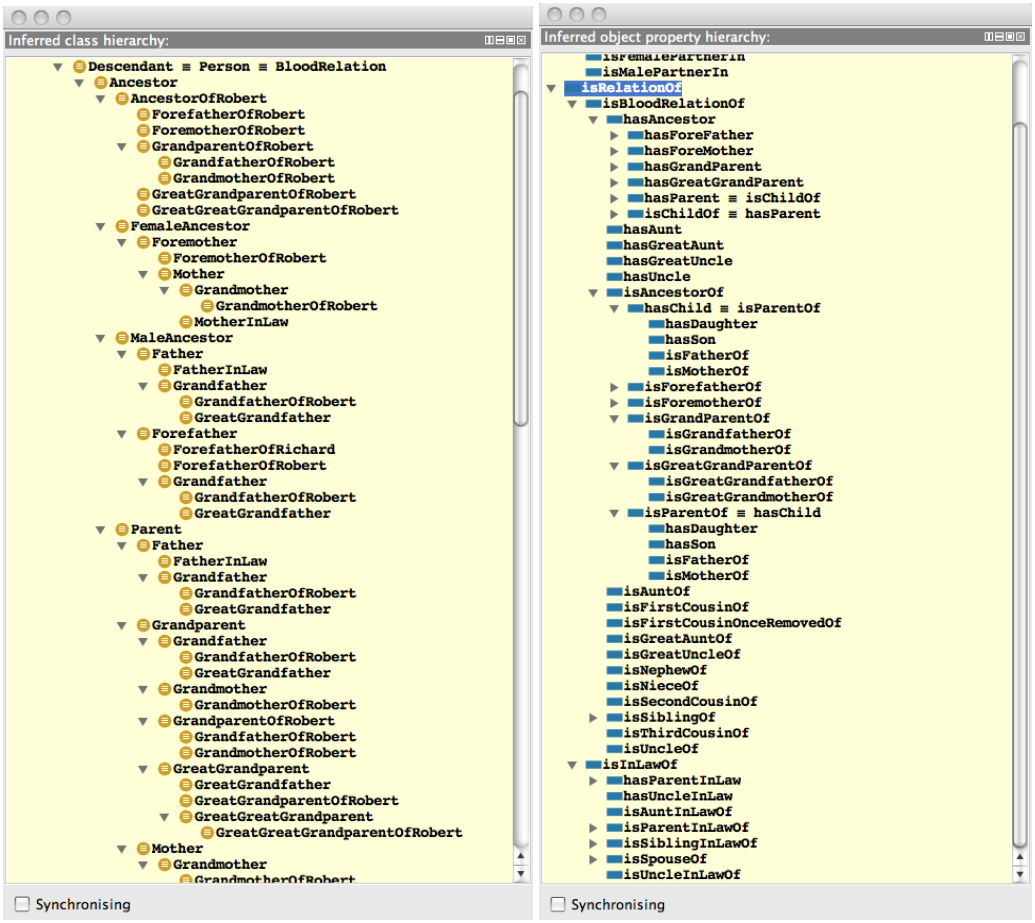

Fig. 1. Class and Property Hierarchies of FHKB

are also entailed from such a hierarchy using sub-property chains, that allows one to declare classes like `Uncle` and finding all uncles in the ontology. The FHKB makes much use of many OWL 2 features and the latest version may be found at `http://www.cs.man.ac.uk/~stevensr/ontology/family.rdf.owl`.

The use of sparse property assertions in the ontology together with rich property hierarchy have several benefits. Firstly, they can keep the ontology small with respect to facts about individuals, while retaining all the necessary entailments. Secondly, while building an ontology using non-sparse facts it is easy to forget some relations, so the knowledge will be incomplete. Thirdly, with many entailments arising from few facts, the ramifications of incorrect facts can be extreme; this may make errors easier to spot and fewer asserted facts make maintenance less onerous. Finally, it is less work for the modeller.

There are, however, some problems in using sparse relations. One of them is that in OWL (2) the ability of reasoning over the properties is much weaker than the ones to reason over classes. Moreover, there are restrictions on how one can use the property axiom in order to obtain a syntactically correct ontology.[1] These issues make it difficult to express some useful facts in OWL 2.

For example, let the ontology O contain the classes `Person`, `Man`, `Woman`, etc. and the properties `isSiblingOf`, `hasBrother` and so on. `Man` and `Woman` are disjoint subclasses of `Person`, and `hasBrother` is a sub-property of `isSiblingOf`. `isSiblingOf` is both symmetric and transitive. Assume that someone needs to express in O the fact that every person that has a male sibling has that sibling as a brother. This example, besides the obvious correspondence to the family ontology, was motivated by a discussion in the OWL Working Group mailing list[2] in 2007. This issue arises regularly when modelling and can, as we will show, enable many entailments.

In the following, we discuss various solutions to the problem and present a new one.

2 Solutions to the Man-Man Problem

A naive approach is to add a GCI that says that every person that has a male sibling necessarily has a brother. In Manchester OWL syntax [3] this would look like:

$$SubClassOf(\texttt{hasSibling some Man}, \texttt{hasBrother some Person}).$$

It is easy to see, however, that this solution is unsatisfactory: in this case, the `hasBrother` property will connect the chosen individual to some *arbitrary* `Person`. I.e., the axiom above together with axioms

$$ObjectPropertyAssertion(\texttt{John}, \texttt{hasSibling}, \texttt{Peter}),$$

$$ClassAssertion(\texttt{Man}, \texttt{Peter})$$

will not entail axiom

$$ObjectPropertyAssertion(\texttt{John}, \texttt{hasBrother}, \texttt{Peter}).$$

Another approach is to use rules and write a rule:

$$\texttt{hasSibling}(x, y), \texttt{Man}(y) \rightarrow \texttt{hasBrother}(x, y).$$

OWL 2, however, has no support for rules; furthermore, in order to keep decidability only DL-safe rules [7] can be used, which allows one to make inferences over individuals, but not over classes. Thus, the axiom

$$SubClassOf(\texttt{isSiblingOf some Father}, \texttt{hasBrother some Person})$$

[1] These restrictions were loosened in [5]; we discuss this later.

[2] `http://lists.w3.org/Archives/Public/public-owl-dev/2007JulSep/0177.html`

can not be inferred in the DL-safe rule system.

The solution proposed in the OWL mailing list was to introduce a new property, `ManMan`, and connect every `Man` to itself by this property:

$$SubClassOf(\texttt{Man}, \texttt{ManMan some Self}).$$

Then add the axiom that the property `isSiblingOf`, when composed with the `ManMan` property, implies `hasBrother`:

$$SubObjectPropertyOf(\texttt{isSiblingOf o ManMan}, \texttt{hasBrother}).$$

In this case, if two individuals are connected via the `isSiblingOf` property, and one of them is a `Man`, then that one has a `ManMan` loop, so the $SubObjectPropertyOf$ axiom will then imply that these two individuals are connected via `hasBrother` property. The correctness of this solution was ensured in the independent publications [1, 6], which propose a rolling-up technique to transform rules into the OWL 2 axioms.

While this approach looks workable, it does have a flaw. The problem is, that after the addition of the $SubObjectPropertyOf$ the axiom, we will introduce an irregularity in the property hierarchy. Indeed, according to [4] the property hierarchy is *regular* if the ordering $\prec$ induced by the property inclusion axioms is acyclic. From the axiom above one can deduce that `isSiblingOf` $\prec$ `hasBrother` and `ManMan` $\prec$ `hasBrother`. As `hasBrother` is a subproperty of `isSiblingOf`, the relation `hasBrother` $\prec$ `isSiblingOf` holds. These implies `isSiblingOf` $\prec$ `hasBrother` $\prec$ `isSiblingOf`. So there is a cycle in the $\prec$ relation, thus the property hierarchy is irregular and the usual reasoning procedures are not then applicable.

Recent research in the area [5] shows that it is possible to reason over *some* ontologies even if their property hierarchy is irregular—namely if it is "stratified". We have good reasons to believe that this approach would allow us to keep our property inclusions, and we would only have to add some additional (possibly even useful) inclusions. To the best of our knowledge, however, none of the OWL 2 reasoners can currently handle stratified property inclusions. Since the modifications to a tableaux reasoner would be non-trivial,[3] we have chosen to pursue an approach that is more direct.

We propose another solution to this problem that also requires a change to current OWL 2 inference engines. Our approach is to introduce a special OWL constructor $Spec(\texttt{R}, \texttt{C})$ for a property `R` and a class `C`, and two tableaux rules that deal with this new constructor. Such a change naturally falls into the tableaux reasoning structure and appears to be straight-forward to implement without affecting the rest of the algorithm. Note that the features of the tableaux algorithm—termination, soundness and completeness— are all preserved in the updated one.

[3] It would involve a novel check for stratified property hierarchies and a different way of constructing certain finite automata used in the tableaux rules

This approach was implemented in the FaCT++ reasoner [9] and evaluated on the FHKB ontology with new axioms.

A new *Property Specialisation* construction *Spec* is introduced to capture the intended expression. This construction comes in two forms, $SpecFrom$ and $SpecInto$, and can be used where the property chain can be used. These two constructions have the following rule semantics:

Syntax	Semantics
$SubObjectPropertyOf(SpecFrom(\mathtt{R}, \mathtt{C}), \mathtt{S})$	$\forall x, y.\ \mathtt{R}(x, y) \wedge \mathtt{C}(x) \rightarrow \mathtt{S}(x, y)$
$SubObjectPropertyOf(SpecInto(\mathtt{R}, \mathtt{C}), \mathtt{S})$	$\forall x, y.\ \mathtt{R}(x, y) \wedge \mathtt{C}(y) \rightarrow \mathtt{S}(x, y)$

Note that having both constructions is unnecessary: one can be expressed via another using inverse properties. Indeed, the axiom

$$SubObjectPropertyOf(SpecInto(\mathtt{R}, \mathtt{C}), \mathtt{S})$$

can be expressed as

$$SubObjectPropertyOf(SpecFrom((\text{ inv } \mathtt{R}), \mathtt{C}), (\text{ inv } \mathtt{S}))$$

according to the semantics. In the following we assume that ontology contains only $SpecFrom$ construction.

In order to reason about an ontology, a *reasoner* is used. Every reasoning task can be reduced to the class satisfiability check that is performed by a reasoner. Various reasoners that perform reasoning for very expressive description logics (including those that underpin OWL 2) use *tableaux* algorithms. These algorithms work by trying to construct a tree-like representation (called a *completion graph*) of a model of the class, starting from an individual instance. Tableaux expansion rules decompose class expressions, add new individuals (e.g., as required by (R some C) terms), and merge existing individuals (e.g., as required by (R max n C) terms). Nondeterminism (e.g., resulting from the expansion of disjunctions) is dealt with by searching various possible models. For an unsatisfiable class, all possible expansions will lead to the discovery of an obvious contradiction known as a *clash* (e.g., an individual that must be an instance of both A and $\neg A$ for some class A); for a satisfiable class, a complete and clash-free model will be constructed [4].

A completion graph is a directed graph $G = (V, E, \mathcal{L})$ where each node $x \in V$ is labelled with a set $\mathcal{L}(x)$ of class expressions and each edge $\langle x, y \rangle \in E$ is labelled with a set $\mathcal{L}(\langle x, y \rangle)$ of property names appear in the ontology. Expansion of the graph is performed according to the set of *completion rules*. Every rule has its own requirements, and if they are satisfied, it fires and change the completion graph. The full set of rules for the DL $\mathcal{SROIQ}$ can be found in [4].

Additional rules for the property specialisation support are shown at Figure 2. The rule *spec*-a-rule does what the semantics of a new construction requires: it checks whether there are two individuals that are connected with the R property and whether the first one is a member of a class C. If so, it also connects them via the S property.

spec-a-rule:	if 1. $SubObjectPropertyOf(SpecFrom(\mathtt{R}, \mathtt{C}), \mathtt{S}) \in O$, 2. x is not a blocked node, $\mathtt{R} \in \mathcal{L}(\langle x, y\rangle), \mathtt{C} \in \mathcal{L}(x), \mathtt{S} \notin \mathcal{L}(\langle x, y\rangle)$ then set $\mathcal{L}(\langle x, y\rangle) = \mathcal{L}(\langle x, y\rangle) \cup \{S\}$
spec-c-rule:	if 1. $SubObjectPropertyOf(SpecFrom(\mathtt{R}, \mathtt{C}), \mathtt{S}) \in O$, 2. x is not a blocked node, $\mathtt{R} \in \mathcal{L}(\langle x, y\rangle), \{\mathtt{C}, \neg\mathtt{C}\} \cap \mathcal{L}(x) = \emptyset$ then set $\mathcal{L}(x) = \mathcal{L}(x) \cup \{\mathtt{D}\}$, where $\mathtt{D} \in \{\mathtt{C}, \neg\mathtt{C}\}$

Fig. 2. Expansion rules for property specialisation

However, this rule alone is not enough for the reasons similar to the QCRs [2]. E.g., assume that class C has a complex definition. Then it might be the case that C is not present itself in the label of node x, but its definition is present there, so it can be inferred that C should be there.

The solution to this problem is the same as the one for the QCRs. If there is a node where *spec*-a-rule can *potentially* be applicable, the special variant of the *choose*-rule, *spec*-c-rule, force the system to determine, whether *spec*-a-rule is applicable or not.

The addition of these rules does not change the correctness of the algorithm. Indeed, similar to other completion rules, it is easy to show that the addition of a new rule keeps the tableaux algorithm terminating, sound and complete.

Moreover, these rules fall nicely in a pay-as-you-go schema: if the ontology contains no rule specialisation axioms, the behaviour of the modified algorithm is exactly the same as the behaviour of the original one.

One thing that should be mentioned is that this new axiom can enforce new subsumptions in the property hierarchy. Besides the trivial case

$$SubObjectPropertyOf(SpecFrom(\mathtt{R}, \mathtt{Thing}), \mathtt{S}),$$

which is easily detectable, and can even be transformed into the axiom

$$SubObjectPropertyOf(\mathtt{R}, \mathtt{S}),$$

there can be more complex cases. It is easy to see, that the set of axioms

$$SubObjectPropertyOf(SpecFrom(\mathtt{R}_1, \mathtt{C}), \mathtt{S})$$

$$SubObjectPropertyOf(SpecFrom(\mathtt{R}_2, \text{ not } \mathtt{C}), \mathtt{S})$$

$$SubObjectPropertyOf(\mathtt{R}_1, \mathtt{R})$$

$$SubObjectPropertyOf(\mathtt{R}_2, \mathtt{R})$$

entails

$$SubObjectPropertyOf(\mathtt{R}, \mathtt{S}).$$

3 Evaluation of the new feature with the FHKB

We use the new approach to reason about the FHKB. The aim was to increase the precision of entailments such as making the general `isSiblingOf` more precise to either `isBrotherOf` or `isSisterOf`. To do this we added the following axioms to the FHKB:

$$SubObjectPropertyOf(SpecFrom(\texttt{isSiblingOf}, \texttt{Man}), \texttt{isBrotherOf});$$

$$SubObjectPropertyOf(SpecFrom(\texttt{isSiblingOf}, \texttt{Woman}), \texttt{isSisterOf});$$

$$SubObjectPropertyOf(SpecFrom(\texttt{isParentOf}, \texttt{Man}), \texttt{isSonOf});$$

$$SubObjectPropertyOf(SpecFrom(\texttt{isParentOf}, \texttt{Woman}), \texttt{isDaughterOf});$$

Note that it is possible to use more axioms, like specialisations for `isFatherOf` and `isMotherOf`. But in the FHKB these 'parental' relations are used as the initial sparse relations in the ABox, so these axioms did not give any extra information. They would help though in the version where the basic sparse property is `isParentOf`, and the gender for every individual in the ABox is given.

We did some tests with this data. All the tests were performed on the 2x Core2Duo Intel Xeon 2.66GHz Mac Pro computer with 16Gb of memory. We used the Protégé4 ontology editor and FaCT++ version 1.3.0 updated to support a new functionality.[4]

Firstly we added these axiom to the FHKB ontology one by one and checked the reasoning time. The results are presented in Table 1. Every value in the table is the arithmetic mean of five runs.

Additional axioms in FHKB	classification time, sec	memory used, Gb
0 (original FHKB)	36.0	0.23
1	69.8	1.0
2	102.0	2.0
3	141.4	3.67
4	173.5	3.67

Table 1. Time increase for classification FHKB with property specialisation axioms

As we can see, in this particular example every addition of such an axiom significantly increases the reasoning time and the necessary memory . One reason for this is the amount of non-determinism that this construction introduces into the KB. In this highly connected graph of Parents and Siblings, the new choose-rule tries to add either Man or not Man to (roughly) every individual; as Man and Woman itself are not primitive, but defined as

$$EquivalentClasses(\texttt{Man}, \texttt{hasGender some Male}),$$

[4] Available at `http://code.google.com/p/factplusplus/`

EquivalentClasses(`Woman`, `hasGender` some `Female`).

the clash detection can be postponed in some cases. Note, however, that even as different axioms interact with each other, there is a linear increase in time.

Another set of tests we did is related to the data discovery. We define a few classes using obvious definitions and count the number of individuals that are instances of these classes. The results are presented in Table 2.

Class	Instances in FHKB	Instances in FHKB +4
`Uncle`	55	76
`GreatUncle`	49	67
`Aunt`	58	77
`GreatAunt`	55	71
`Brother`	160	160
`isBrotherOf` some `Person`	152	160
`hasBrother` some `Person`	128	257
`Sister`	163	163
`isSisterOf` some `Person`	153	163
`hasSister` some `Person`	115	251
`BrotherInLaw`	39	39
`isBrotherInLawOf` some `Person`	25	39
`hasBrotherInLaw` some `Person`	37	38
`SisterInLaw`	30	30
`isSisterInLawOf` some `Person`	12	30
`hasSisterInLaw` some `Person`	33	37
`isSonOf` some `Person`	7	202
`isDaughterOf` some `Person`	6	208
`hasSon` some `Person`	7	122
`hasDaughter` some `Person`	5	119

Table 2. Number of instances for some classes in FHKB

Additional tableaux rules force some siblings to specialise into brothers or sisters, thus the increased number of instances in the appropriate classes.

We can found some interesting information out of these experiments. One observation is that after adding the extra rules, classes represented by the class expressions (`isSonOf` some `Person`) and (`isDaughterOf` some `Person`) became equivalent to `Man` and `Woman` respectively. This happens as, e.g. `Man` is defined as a `Person` (with additional restrictions), and every person have a parent, thus (according to the new rule) they are also connected to their parents via `isSonOf` property.

Another example is an increased number of siblings-in-law. This were used to find a bug in modelling in the FHKB. At some point it appears that the number of instances of the class (`isBrotherInLawOf` some `Person`) were larger than the number of instances of `BrotherInLaw`. After the investigation we found out that `isBrotherInLawOf` counts the siblings of a spouse as well as the spouses of a

sibling, while `BrotherInLaw` counts only the former. After that oversight was fixed, the number of instances appears to be the same, as expected.

4 Discussion

We have proposed a new solution for the recurring 'Man-Man' problem. This solution, despite it requires a syntax extension to OWL and extending tableaux rules of the reasoning algorithms, appears to be a natural extension of OWL 2. The extension has been implemented in the FaCT++ reasoner.

Evaluation of the new feature on the (extended) FHKB shows definite benefits. The sparse assertion of, for instance, sibling relationships becomes a feasible modelling approach. The cost of such an extension has been shown to be low.

The Man-Man issue is not the only one to arise from the FHKB. As described in Stevens and Stevens [8], all entailments on individuals are correct, bar one kind. This was the attempt to use sub-property chains to determine cousin relationships. The obvious sub-property chain is `hasParent` o `isSiblingOf` o `isParentOf`. The property `isSiblingOf` is symmetric and transitive. Hence it cannot be made irreflexive because it would make cause an inconsistency. This means that an individual is its own sibling; thus, that individual and his siblings are his or her own cousins.

Another aspect of family relationships that cannot be determined using OWL 2 and automated reasoning are half-siblings and step-parentage. At the level of individuals, such relationships can be determined using DL-safe rules. These are, however, restricted to use in the ABox.

The FHKB has been developed as a tutorial example.[5] It highlights many of OWL 2's features, as well as emphasising individuals; entailments on individuals; property hierarchies and property characteristics. As well as exhibiting some of the limitations of OWL 2 as outlined above, it highlights the issue of scalability of the application of sophisticated reasoning over ABoxes. The FHKB also highlights some traps for the unwary modeller using OWL:

– It is tempting having made sub-property chains such as `isUncleOf` and `isGrandparentOf`, to make class definitions such as

$$EquivalentTo(\texttt{Uncle}, \texttt{Man} \text{ that } \texttt{isUncleOf} \text{ some } \texttt{Person}).$$

As a consequence, all known uncles are indeed inferred to be instances of `Uncle`. It does not, however, have the desired effects when building a TBox. All uncles are necessarily brothers and so the uncle class should be subsumed by the class `Brother`; this is not the case, because

$$SubObjectPropertyOf(\texttt{isBrotherOf} \circ \texttt{isParentOf}, \texttt{isUncleOf})$$

is not

$$EquivalentObjectProperties(\texttt{isBrotherOf} \circ \texttt{isParentOf}, \texttt{isUncleOf}),$$

which can not be expressed in OWL 2.

[5] `http://www.cs.man.ac.uk/~stevensr/menupages/fhkb.php`

– We could be tempted to expect that the creation of the defined class for the uncles of `robert_david_bright_1965` and `richard_john_bright_1962` should be inferred to be equivalent classes as they both contain the same individuals. Yet this is not the case for the similar reason as the first case: Robert could, in principle, have an uncle who is *not* Richard's uncle.

The FHKB serves as a good example for OWL 2 features; both as a tutorial example and for testing reasoners. For teaching, it touches on a majority of the issues encountered when using OWL. It also highlights limitations in the expressivity of OWL. As demonstrated with the projection case study presented in this paper, it can act as a test-bed for extensions to OWL.

References

1. Francis Gasse, Ulrike Sattler, and Volker Haarslev. Rewriting Rules into SROIQ Axioms. In *Proc. of the 21st International Workshop on Description Logics (DL2008), Dresden, Germany, May 13-16, 2008*, 2008.
2. B. Hollunder and F. Baader. Qualifying Number Restrictions in Concept Languages. In *Proc. of the Second International Conference on Principles of Knowledge Representation and Reasoning, KR-91*, pages 335–346, Boston (USA), 1991.
3. Matthew Horridge and Peter F. Patel-Schneider. Manchester OWL Syntax for OWL 1.1. In *Proc. of OWL: Experiences and Directions Workshop (OWLED 2008DC)*, 2008.
4. Ian Horrocks, Oliver Kutz, and Ulrike Sattler. The Even More Irresistible $\mathcal{SROIQ}$. In *Proc. of the 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2006)*, pages 57–67. AAAI Press, 2006.
5. Yevgeny Kazakov. An Extension of Regularity Conditions for Complex Role Inclusion Axioms. In *Proc. of the 2009 Description Logic Workshop (DL 2009)*, 2009.
6. Markus Krötzsch, Sebastian Rudolph, and Pascal Hitzler. Description Logic Rules. In Malik Ghallab, Constantine D. Spyropoulos, Nikos Fakotakis, and Nikos Avouris, editors, *Proc. of the 18th European Conference on Artificial Intelligence (ECAI-08)*, volume 178 of *Frontiers in Artificial Intelligence and Applications (FAIA)*, pages 80–84. IOS Press, Jul 2008.
7. Boris Motik, Ulrike Sattler, and Rudi Studer. Query Answering for OWL-DL with Rules. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, 3(1):41–60, 2005.
8. Robert Stevens and Margaret Stevens. A Family History Knowledge Base Using OWL 2. In *Proc. of OWL: Experiences and Directions Workshop (OWLED 2008)*, 2008.
9. Dmitry Tsarkov and Ian Horrocks. FaCT++ description logic reasoner: System description. In *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2006)*, volume 4130 of *Lecture Notes in Artificial Intelligence*, pages 292–297. Springer, 2006.