Reconciling OWL and Rules

David Carral Martínez, Adila Krisnadhi, Frederick Maier, Kunal Sengupta, and Pascal Hitzler

Kno.e.sis Center, Wright State University, Dayton OH 45435, USA

Abstract. We report on a recent advance in integrating Rules and OWL. We discuss a recent proposal, known as *nominal schemas*, which realizes a seamless integration of Datalog rules into the description logic $\mathcal{SROIQ}$ which underlies OWL 2 DL. We present extensions of the standardized OWL syntaxes to incorporate nominal schemas, reasoning algorithms, and a first naive implementation. And we argue why this approach goes a long way towards overcoming the present paradigm split.

Keywords: OWL, description logic, rules, datalog, decidability, reasoning algorithms, local closed world

1 Introduction

The desire to integrate the Web Ontology Language OWL [28] with rule-based knowledge representation formalisms has produced a significant number of research contributions on the topic. The importance of the research is probably best understood by looking at the current version of the Semantic Web Layer Cake.[1] It shows OWL as an alternative to the Rule Interchange Format RIF [15], with a *unifying logic* as an overarching formalism integrating them both. Some of the past proposals for such a unifying logic have significant theoretical appeal, and some of them have also been used effectively in applications, but the Semantic Web research community has yet to form a consensus on the matter.

Why past and current proposals for a unifying logic have not yet received wide support is not an easy question to answer. There are likely many reasons. Some of the following issues, however, appear to be relevant. First, some proposed unifying logics make use of a *hybrid syntax*, one combining OWL syntax (i.e. a syntax which focuses on classes and their relationships using class subsumption axioms) with a distinct rules syntax. In this way, a knowledge base consists of two different parts, each part fitting one of the two paradigms. Second, some of the proposals make use of a complicated (and sometimes also hybrid) semantics. A straightforward extension of semantics would clearly be preferred. Third, the more straightforward combinations of OWL and rules are *undecidable*. At least from the perspective of OWL design criteria, this is highly undesirable.

In this paper, we discuss a recently made proposal for extending OWL with rules [21] and suggest that it goes some way toward providing a unifying logic

[1] http://www.w3.org/2007/03/layerCake.svg

that does not suffer from the above mentioned drawbacks. That is, it uses a single syntax and semantics, and it is decidable. The proposal is based on extending description logics with *nominal schemas*, which are expressions akin to the variables of DL-safe rules. In [21], the following theoretical results are shown:

- $\mathcal{SROIQ}$, which forms the basis for OWL, remains decidable when extended with nominal schemas. Indeed, the worst case computational complexity does not increase.
- Tractable profiles extended with nominal schemas remain tractable.
- Binary Datalog—with the safety restriction that variables refer to named individuals—is subsumed by the approach, in the sense that the rules can be translated into an equivalent DL theory making use of nominal schemas.

Furthermore, the semantics is a straightforward and intuitively easy to understand extension of DL semantics. The proposal is *OWL-centric* in that it both attempts to provide a unifying logic from the perspective of design rationales of OWL and also builds on the DL syntax and semantics underlying OWL.

The purpose of this paper is to add further support to the claim that the proposed formalism of [21] provides a promising step towards unifying OWL and rules. Specifically:

1. We provide extensions of the different OWL syntaxes defined by the OWL 2 specification [10].
2. We report on a naive reasoning implementation and performance evaluations, which indicate that the OWL extension is computationally feasible.
3. We show how the proposed approach covers all of positive Datalog (i.e., not only the binary fragment), which forms the basis for RIF Core [2].
4. We show how existing algorithms for description logics around OWL can be extended with relative ease to incorporate nominal schemas.
5. We provide first pointers on how to extend our approach to cover non-monotonic rules.

The plan of the paper is as follows. In Section 2 we recall earlier results on nominal schemas and show how we can capture Datalog with this notion. In Section 3 we show how the different OWL 2 syntaxes can easily be extended to cover nominal schemas. In Section 4 we report on a first straightforward implementation of reasoning with nominal schemas, and on an evaluation of it. In Section 5, we show how existing tableau algorithms can be enhanced for reasoning with nominal schemas. In Section 6 we discuss the potential of our approach to cover rules beyond monotonic Datalog. In Section 7, we conclude and discuss open issues. Supplementary material, including our implementation and test data, is available from http://wiki.knoesis.org/index.php/NSByGrounding.

2 Integrating OWL and Rules Through Nominal Schema

We briefly recall the technical material from [21], and show how nominal schemas make it possible to incorporate Datalog. We will refrain from a fully formal presentation (which can be found in [17, 21]), and instead focus on readability and intuitive understanding.

From a description logic (DL) perspective, a nominal schema is a kind of variable nominal (a term used in [20]). I.e., a nominal schema can be used at any place where a nominal is allowed in the DL under consideration, the difference being that a nominal schema carries a variable identifier instead of an individual name. This exact variable identifier can then also appear in other places of the same axiom, effectively resulting in a binding between these occurrences.

Consider the following example from [17]. It states that somebody has a conflicting review assignment (paper x) if this person has a paper submitted at the same event z which is co-authored by one of the authors y of paper x. Each of $\{x\}$, $\{y\}$, and $\{z\}$ is a nominal schema, occurring multiple times in the axiom.

$$\begin{aligned} &\exists \text{hasReviewAssignment}.((\{x\} \sqcap \exists \text{hasAuthor}.\{y\}) \sqcap (\{x\} \sqcap \exists \text{atVenue}.\{z\})) \\ &\quad \sqcap \exists \text{hasSubmittedPaper}.(\exists \text{hasAuthor}.\{y\} \sqcap \exists \text{atVenue}.\{z\}) \\ &\quad \sqsubseteq \exists \text{hasConflictingAssignedPaper}.\{x\} \end{aligned} \tag{1}$$

Semantically, nominal schemas can be bound only to nominals, i.e., a class indicated by a nominal schema can contain only known (named) individuals from the knowledge base. As such, they are reminiscent of DL-safe rules [27] or more precisely of DL-safe variables [19, 20]. The formal semantics has been spelled out explicitly in [21], however the same semantics can be obtained by a transformational approach which eliminates all nominal schemas by *full grounding*: We replace an axiom with nominal schemas by all axioms (without nominal schemas) which can be obtained by substituting all nominal schemas by nominals. The above example (1) thus gives rise to all axioms of the form

$$\begin{aligned} &\exists \text{hasReviewAssignment}.((\{a_i\} \sqcap \exists \text{hasAuthor}.\{a_j\}) \sqcap (\{a_i\} \sqcap \exists \text{atVenue}.\{a_k\})) \\ &\quad \sqcap \exists \text{hasSubmittedPaper}.(\exists \text{hasAuthor}.\{a_j\} \sqcap \exists \text{atVenue}.\{a_k\}) \\ &\quad \sqsubseteq \exists \text{hasConflictingAssignedPaper}.\{a_i\} \end{aligned}$$

where a_i, a_j and a_k are individual names. The resulting knowledge base carries the classical DL semantics.

We indicate the addition of the nominal schema construct to a DL by the letter $\mathcal{V}$, following established practice. Thus, the DL $\mathcal{SROIQ}$ extended with nominal schemas is called $\mathcal{SROIQV}$. It was shown in [21] that $\mathcal{SROIQV}$ has the same worst-case computational complexity as $\mathcal{SROIQ}$, and that any binary Datalog rule can be encoded into $\mathcal{SROIQV}$ in such a way that ground entailments are preserved. In fact, it was shown that $\mathcal{SROELV}$ suffices for this.[2]

Naive grounding results in an increase in the number of axioms which is worst-case exponential in the number of named individuals in the knowledge base. If k is the number of named individuals in the knowledge base, then axiom (1) results in k^3 axioms without nominal schemas—the number 3 comes from the fact that the axiom contains 3 distinct nominal schemas. We come back to this in Section 4.

[2] $\mathcal{SROEL}$ [18, 20] is essentially the tractable (polynomial) OWL 2 EL profile [25].

It was shown in [21] (with a slight generalization in [17]) that a variant of $\mathcal{SROELV}$ can be obtained, called $\mathcal{SROELV}_n$,[3] which results only in a polynomial number of new axioms, provided a method of *smart grounding* is used. For details, please refer to [17, 21]. For the above example axiom (1), we indeed obtain only $k + 2k^2$ new axioms through smart grounding [17]. We will again come back to this in Section 4.

Capturing Datalog with Nominal Schemas The method for capturing binary Datalog in $\mathcal{SROELV}$ from [21] does not carry over directly to general Datalog, and thus requires modifications, resulting in the following.

Given a Datalog rule $A_1, \ldots, A_n \rightarrow A$, where A and all A_i are atomic formulas of the form $p(x_1, \ldots, x_n)$ with the x_i being variables, we translate this rule into the DL axiom $\tau(A_1) \sqcap \cdots \sqcap \tau(A_n) \sqsubseteq \tau(A)$. For an atomic formula $p(x_1, \ldots, x_n)$, we define $\tau(p(x_1, \ldots, x_n))$ to be the DL class expression $\exists U.(\exists p_1.\{x_1\} \sqcap \cdots \sqcap \exists p_n.\{x_n\})$, where U is the universal role and $p_1, \ldots, p_n$ are role names used exclusively for encoding occurrences of the n-ary predicate symbol p. If x_i is a constant, then the corresponding nominal schema becomes a nominal. Due to space constraints, we omit the proof of the following theorem.[4]

Theorem 1. *The transformation just described converts a set P of Datalog rules into a $\mathcal{SROELV}$ knowledge base K, such that, for any n-ary predicate symbol p in P and any n-tuple $(a_1, \ldots, a_n)$ of constants in P, we have that $P \models p(a_1, \ldots, a_n)$ if and only if $K \models \top \sqsubseteq \exists U.(\exists p_1.\{a_1\} \sqcap \cdots \sqcap \exists p_n.\{a_n\})$.*

3 OWL Syntax

We now show how to represent nominal schemas in the main syntaxes of OWL 2 [28].[5] New reserved words are presented to mark the appearance of nominal schemas in the different syntaxes. Several approaches were considered for the representation and storage of nominal schemas, such as the use of entities within the ontology namespace, but this paper proposes the use of string literals. Using this approach we prevent the possible overlap that could be produced by giving the same name to two different nominal schemas. If these are declared as entities and, by error, two of them share the same name they will end up pointing to the same node in an RDF graph when they most likely refer to different individuals.

The selected approach using the xsd:string datatype is also used in the RIF XML and RDF syntaxes [3, 15]. Note that the same nominal schema can never

[3] n is a natural number which serves as a kind of global bound on the number of different nominal schemas in an axiom—however the exact definition of $\mathcal{SROELV}_n$ is more general, see [17, 21].

[4] A full proof can be found in the technical report accompanying this paper, which is available from http://wiki.knoesis.org/index.php/NSByGrounding.

[5] Mapping from Turtle triples to RDF/XML is well defined and so the RDF/XML based syntax will not be directly addressed.

appear in two different statements of an ontology.[6] A nominal schema will only be related with one single axiom. By using a string type the occurrence of the nominal schema is exclusively bound to the axiom where it appears and the same string could be repeated in different axioms along the ontology safely. Even if two nominal schemas use the same string it will be considered as different occurrences of a datatype and therefore, they will be two separated nodes in an RDF graph.

Using underscores to mark the appearance of a nominal schema, as for Turtle blank nodes, was also considered. We rejected this because it could induce errors. Although in some cases both nominal schemas and blank nodes can represent individuals in an RDF graph they are completely different concepts. Using the underscore to mark both could be tricky and would make mappings from and to Turtle syntax difficult to define. With such a similar syntax the mapping may produce errors confusing nominal schemas with blank nodes and problems may arise when we want to move from the Turtle syntax to an RDF Graph.

Functional Syntax The required modifications for the Functional Syntax grammar [26] are very minimal. The reserved word ObjectVariable is used to mark the appearance of the nominal schema. The nominal schema variable name will be in parentheses be followed by the expression '^^xsd:string'.

To realize this, we need to make only two changes to the grammar in [26].

1. Add **ObjectVariable** as alternative to the **ClassExpression** rule:

$$\textbf{ClassExpression} := \textbf{Class} \mid \textbf{ObjectIntersectionOf} \mid \ldots \mid \textbf{DataExactCardinality} \mid \textbf{ObjectVariable}$$

2. Add the following production rule to the grammar.

$$\textbf{ObjectVariable} := \text{'ObjectVariable('}\textbf{quotedString}\text{'^^xsd:string)'}$$

Although it can be argued whether, conceptually, nominal schemas are class expressions, their addition in this part of the grammar has been chosen in order to keep modifications as small as possible.

Manchester Syntax Again, the reserved word ObjectVariable will be used to mark the appearance of nominal schemas in the Manchester Syntax [13]. As in the Functional Syntax, the nominal schema variable will be in parentheses and followed by '^^xsd:string'. The needed changes to this grammar are the following.

1. Add **ObjectVariable** as alternative to the **atomic** rule:

$$\textbf{atomic} := \textbf{classIRI} \mid \text{'\{'}\textbf{individualList}\text{'\}'} \mid \text{'('}\textbf{description}\text{')'} \mid \textbf{ObjectVariable}$$

2. Add the following production rule to the grammar.

$$\textbf{ObjectVariable} := \text{'ObjectVariable('}\textbf{quotedString}\text{'^^xsd:string)'}$$

[6] Put differently, if a nominal schema appears in two different axioms, then the occurrences need to be considered distinct.

Turtle RDF Syntax We define the syntax of nominal schemas in Turtle [1] through a mapping from the Functional Syntax to the triple-notation, as in [29]. To add nominal schemas syntax to the mappings add the following row to the mapping from the Functional Syntax to Turtle as specified in [29].

Functional-Style Syntax	Triples Generated	Main Node
ObjectVariable("v1"^^xsd:string)	_:x rdf:type owl:ObjectVariable _:x owl:variableId "v1"	_:x

For the converse direction, add the following row to the mapping from Turtle to the Functional Syntax specified in [29].

RDF/XML Triples	Functional Syntax
_:x rdf:type owl:ObjectVariable _:x owl:variableId "v1"	ObjectVariable("v1"^^xsd:string)

4 A Reasoning Implementation via Grounding

We have realized an implementation of nominal schema reasoning using naive and smart grounding (see Section 2), primarily to obtain a baseline for the development and testing of more efficient algorithms, and in order to show that even the grounding approach can be used for small use cases or for initial testing. We utilize Pellet [31] and the OWL API [12] for implementation and experiments.

For the experiments we selected several ontologies from the TONES repository[7] and added axioms with nominal schemas. We were particularly interested in exploring the limit of usefulness of the grounding approach, and therefore varied the number of axioms with nominal schemas as well as the number of nominal schemas per axiom. Reasoning times (using Pellet after grounding) are averaged over 100 runs, and load time is reported separately. The reasoning task used in the experiments was satisfiability checking. We also report on experiments using smart grounding, and on casting OWL RL into $\mathcal{SROELV}n$.

Testing was performed using a 64-bit Windows 7 computer with an Intel(R) Core(TM) i5 CPU processor. A Java JDK 1.5 version was used allocating 3GB as the minimum for the Java heap and 3.5GB as the maximum for each experiment. The implementation and test data, and some additional evaluation data, are available from http://wiki.knoesis.org/index.php/NSByGrounding.

Naive Grounding We selected seven ontologies from the TONES repository with different numbers of individuals for this evaluation. Some of them were slightly modified, e.g., by randomly populating them with individuals. Basic metrics are presented in Table 1.

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

Ont	Ind	Classes	Ann	Data	Obj	no ns		1 ns		2 ns		3 ns	
Fam[8]	5	4	0	1	11	0.01	0.00	0.01	0.00	0.01	0.00	0.04	0.02
Swe[9]	22	189	1	6	25	3.58	0.08	3.73	0.07	3.85	0.10	10.86	1.11
Bui[10]	42	686	15	0	24	1.70	0.16	1.50	0.15	2.75	0.26	74.00	6.68
Wor[11]	80	1842	6	0	31	0.11	0.04	0.12	0.05	1.10	0.55	*11,832.00	*315.00
Tra[12]	183	445	2	4	89	0.05	0.03	0.05	0.02	5.66	1.76	OOM	OOM
FTr[13]	368	22	2	6	52	0.03	4.28	0.05	5.32	35.53	42.73	OOM	OOM
Eco[14]	482	339	2	8	45	0.04	0.24	0.07	0.02	56.59	13.67	OOM	OOM

Table 1. Ontologies used in experiments for naive grounding and naive grounding experimental results. Ind: individuals, Ann: Annotation Properties, Data: Data Properties, Obj: Object Properties. For the remaining entries, the first listed number is load time, the second is reasoning time, both in seconds. * indicates approximate values, we used only 5 runs for these. OOM indicates *out of memory*.

In order to understand the effect of several nominal schemas on the runtime, we added three different types of axioms to the ontologies, (1) an axiom with only one nominal schema, (2) an axiom with two different nominal schemas, and (3) an axiom with three different nominal schemas. An example for an added axiom is

$$\exists \text{prop1}.\{v1\} \sqcap \exists \text{prop2}.\{v1\} \sqcap \exists \text{prop3}.\{v2\} \sqcap \exists \text{prop4}.\{v2\} \sqsubseteq \text{Class1}.$$

Since the blow-up obtained from naive grounding is exponential in the number of nominal schemas, this is already the limit we can manage with non-trivial ontologies. We will discuss the impact of this further below. Results are presented in Table 1.

We then investigated the impact of *several* axioms with nominal schemas on the performance, by adding 20 axioms with one nominal schema, respectively 10 axioms with 2 nominal schemas. The results can be found in Table 2.

Discussion First of all, it must be noted that grounding is obviously a rather stupid approach to reasoning with nominal schemas. We provide these figures as a baseline, and not because we think that reasoning with nominal schemas should be done this way. Rather, algorithms need to be developed which perform grounding *by need* and in an incremental manner, and we are going to further discuss this in Section 5. Nevertheless, the figures given above show that even the

[8] http://www.mindswap.org/ontologies/family.owl
[9] http://sweet.jpl.nasa.gov/1.1/data.owl
[10] http://www.ordnancesurvey.co.uk/ontology/BuildingsAndPlaces/v1.1/BuildingsAndPlaces.owl
[11] http://www.berkeleybop.org/ontologies/obo-all/worm_phenotype_xp/worm_phenotype_xp.obo
[12] http://reliant.teknowledge.com/DAML/Transportation.owl
[13] http://www.co-ode.org/roberts/family-tree.owl
[14] http://reliant.teknowledge.com/DAML/Economy.owl

Ontology	Individuals	no ns		20×1 ns		10×2 ns	
Fam	5	0.01	0.00	0.01	0.00	0.02	0.01
Swe	22	3.58	0.08	3.42	0.08	3.73	0.28
Bui	42	2.70	0.16	2.69	0.25	5.70	3.21
Wor	80	0.11	0.04	0.23	0.28	12.42	6.88
Tra	183	0.05	0.03	0.33	0.15	107.57	43.63
FTr	368	0.03	4.28	0.52	11.33	OOM	OOM
Eco	482	0.04	0.24	0.65	0.30	OOM	OOM

Table 2. More naive grounding experimental results, the first listed number is load time, the second is reasoning time, both in seconds. OOM indicates *out of memory*.

naive grounding approach is not entirely hopeless for small prototype ontologies, if the number of axioms requiring nominal schemas remains small, and there are not more than two different nominal schemas per axiom.

To understand the expressivity of axioms with two different nominal schemas, note that such an axiom expresses the knowledge which could otherwise only be stated using k^2 axioms without nominal schemas, where k is the number of individual names used in the knowledge base. This suffices, for example, to encode all of OWL RL (but for functionality axioms) in $\mathcal{SROELV}$.

Smart Grounding For completeness of our evaluation, we also tested the smart grounding approach (see Section 2). We chose different test ontologies for this purpose since they need to lie in the OWL 2 EL profile for this approach. The ontologies are listed below; individuals were artificially (randomly) added since OWL EL ontologies in the TONES repository do usually not sport individuals.

Ontology	Classes	Annotation P.	Data P.	Object P.	Individuals
Rex[15]	552	10	0	6	100
Spatial[16]	106	13	0	13	100
Xenopus[17]	710	19	0	5	100

As before, we added three different types of axioms to the ontologies: (1) an axiom with only one nominal schema, (2) an axiom with two different nominal schemas, and (3) an axiom with three different nominal schemas. All occurrences of nominal schemas were *safe* as specified in the smart grounding approach detailed in [17, 21].[18] The results of the evaluation can be found in Table 3.

[15] http://obo.cvs.sourceforge.net/*checkout*/obo/obo/ontology/physicochemical/rex.obo

[16] http://obo.cvs.sourceforge.net/*checkout*/obo/obo/ontology/anatomy/caro/ spatial.obo

[17] http://obo.cvs.sourceforge.net/*checkout*/obo/obo/ontology/anatomy/gross_anatomy/animal_gross_anatomy/frog/xenopus_anatomy.obo

[18] For example, the first occurrence of each of the nominal schemas $\{y\}$ and $\{z\}$ in axiom (1) is safe, which suffices for smart grounding.

Ontology	Individuals	No ns		1 ns		2 ns		3 ns	
Rex naive	100	0.025	0.009	0.031	0.013	1.689	0.112	OOM	OOM
Rex smart	100			0.058	0.023	0.046	0.011	0.053	0.009
Spatial naive	100	0.035	0.029	0.021	0.014	1.536	0.101	OOM	OOM
Spatial smart	100			0.018	0.013	0.033	0.007	0.044	0.011
Xenopus naive	100	0.063	0.018	0.070	0.190	1.598	0.112	OOM	OOM
Xenopus smart	100			0.099	0.037	0.083	0.018	0.097	0.063

Table 3. Evaluation of the smart grounding approach. The "No ns" column of the table refers to the time when no axiom was added to the ontology. Each one of the next pair of columns presents the results for each added axiom. The first column of each pair in the table refers to loading time and the second one shows the time for checking knowledge base satisfiability. Each ontology is repeated in two rows showing the difference of time between the smart grounding approach and the naive grounding approach.

Discussion The evaluation shows rather clearly that smart grounding significantly improves efficiency, i.e., smart grounding appears to scale well if occurrences of nominal schemas are safe, in which case the smart grounding approach yields a clear advantage. Since nominal schemas are rather new, it remains to be investigated to which extent the use of safe occurrences of nominal schemas already provides added value to practice.

Casting OWL RL to $\mathcal{SROELV}_n$ We finally investigated the particular task of encoding OWL RL ontologies [25] in $\mathcal{SROELV}_n$. There are two reasonable ways how to do this. The first interprets OWL RL ontologies (after the usual normalization) as DL-safe [27] binary datalog rules with equality (i.e., such that ground entailments are preserved), and then transforms these rules into $\mathcal{SROELV}_n$, as indicated in [21].[19] When this is done, then all but one type of OWL RL axioms are actually translated into $\mathcal{SROELV}_2$ axioms, as each of these axioms carries at most two different nominal schemas. We have already seen earlier in this section, that even the naive grounding approach scales rather well in this case. The single type of axiom which requires three nominal schemas in this transformation is the functionality declaration of roles.

In the following evaluation, we take a different approach. Any OWL RL axiom (after normalization) which can be expressed in $\mathcal{SROEL}$ (i.e., in OWL EL) is left unmodified. However, not all OWL RL axioms are part of the OWL EL language, so we have to find alternatives for those axioms.

For role inverses, i.e. for axioms of the form $R^- \sqsubseteq S$, we interpret it as the DL-safe rule $R(x, y) \rightarrow S(y, x)$, encoded in $\mathcal{SROELV}_n$ as the axiom

$$\exists U.(\{x\} \sqcap \exists R.\{y\}) \sqsubseteq \exists U.(\{x\} \sqcap \exists S.\{y\})).$$

[19] The transformation from Section 2 can be used, but the simpler one from [21] suffices.

Our test ontology[20] has 39 classes, 8 annotation properties, no data properties, 15 object properties, and 100 individuals. We use up to five role inverses. The following table shows the runtimes in seconds, as usual giving first the load time and then the reasoning time.

	1 inverse		3 inverses		5 inverses	
OWL RL ontology	0.51	0.00	0.397	0.00	0.61	0.00
translation	3.53	0.46	10.59	1.93	22.03	4.48

For the universal quantifier, i.e. for axioms of the form $A \sqsubseteq \forall R.C$, we interpret it as the DL-safe rule $A(x) \wedge R(x, y) \rightarrow C(y)$, which can be expressed in $\mathcal{SROELV}_n$ as the axiom

$$\exists U.(\{x\} \sqcap A) \sqcap \exists U.(\{x\} \sqcap \exists R.\{y\}) \sqsubseteq \exists U.(\{y\} \sqcap C).$$

Our test ontology[21] has 96 classes, 8 annotation properties, 32 data properties, 60 object properties and 100 individuals, and was evaluated similarly to the case for role inverse, with the following results.

	1 universals		3 universals		5 universals	
OWL RL ontology	0.07	0.00	0.01	0.00	0.02	0.00
translation	1.88	0.36	6.47	0.90	13.85	1.68

Discussion While the increase in runtime is significant, it appears that this translation can be used for moderate-sized ontologies, even with the naive grounding approach. At first sight, the casting of OWL RL ontologies into OWL EL ontologies seems to be a rather dubious undertaking. However, it can be useful if an OWL RL ontology is to be used alongside an OWL EL ontology, as reasoning in $\mathcal{SROELV}_n$, which contains both profiles using the above transformation, is tractable, while a naive combination of the two profiles is not.

5 Dealing with Nominal Schemas in Existing Algorithms

As mentioned in Section 4, reasoning with nominal schemas can be done via grounding the nominal schemas into known individuals in the knowledge base. However, both naive grounding and smart grounding essentially do the grounding up front as a preprocessing phase before the knowledge base satisfiability is computed. It has been noted in the previous section, also in [21], that one possible optimization for these approaches is by treating nominal schemas like nominals in the reasoning process, and then grounding them with concrete individuals only if this is needed and relevant for deduction steps. In this section, we briefly sketch a way to realize this *delayed* or *deferred grounding*.

We consider adding a delayed grounding rule to a tableau-based algorithm for deciding concept satisfiability w.r.t. a knowledge base. As an example, we

[20] Modified from http://ontology.dumontierlab.com/pharmacogenomics-primitive.
[21] Modified from http://org.semanticweb.ontologies/Ontology1225725433367251000.

consider the tableau algorithm for $\mathcal{SROIQ}$ [14]. Actually, that tableau algorithm decides concept satisfiability w.r.t. reduced RBox only, while the TBox and ABox components of the knowledge base are internalised as part of the input concept description. Let C_0 be an input concept and $\mathcal{R}$ be a reduced RBox input. Assume that $o_i = \{a_i\}$, $1 \leq i \leq \ell$ are nominals in C_0. The algorithm starts with a set of tableau nodes $\{s_0, s_1 \ldots, s_\ell\}$ where $\mathsf{L}(s_0) = \{C_0\}$, and $\mathsf{L}(s_i) = \{o_i\}$, $1 \leq i \leq \ell$. Here, $\mathsf{L}(s)$ is the label of node s which is the set of concepts associated with s. Intuitively, the structure of the tableau gives off a model of C_0 and $\mathsf{L}(s)$ contains all concepts to which the individual represented by the node s must belong. The algorithm proceeds by applying *tableau expansion rules* whose details can be seen in [14], until no rule is applicable or a clash occurs. If a clash occurs, then C_0 is not satisfiable w.r.t. $\mathcal{R}$.

The DL $\mathcal{SROIQV}$ is obtained from adding nominal schemas to the DL $\mathcal{SROIQ}$. Hence, given a $\mathcal{SROIQV}$ knowledge base, a naive grounding of nominal schemas will yield a knowledge base in $\mathcal{SROIQ}$ whose satisfiability can then be decided using the tableau algorithm mentioned above. We specify a grounding rule that enables grounding to be done during tableau expansion instead of doing it up front. For this rule, we first need to define some needed notions. For every concept C, let $\mathsf{Var}(C)$ be the set of all variables appearing as a nominal schema in C. We say that two concepts C_1 and C_2 are *pairwise connected* if $\mathsf{Var}(C_1) \cap \mathsf{Var}(C_2) \neq \emptyset$. A concept C is *directly connected* if either $C = C_1 \sqcap C_2$ or $C = C_1 \sqcup C_2$ where C_1 and C_2 are pairwise connected. A concept C is *immediately ns-propagating* if C is in one of the following forms: $\exists R.D$, $\forall R.D$, $(\leq n\, R.D)$, and $(\geq n\, R.D)$, such that $D = \{x\} \sqcap D'$, $\neg\{x\} \sqcap D'$, $D = \{x\} \sqcup D'$, or $D = \neg\{x\} \sqcup D'$ for some nominal schema $\{x\}$. The grounding rule is as follows.

$$\begin{aligned} \textsf{grounding}: \quad & \text{if } C \in \mathsf{L}(s), \{z\} \text{ is a nominal schema in } C, \\ & \quad C[z/a_i] \notin \mathsf{L}(s) \text{ for some } i, 1 \leq i \leq \ell \\ & \text{then } \mathsf{L}(s) := \mathsf{L}(s) \cup \{C[z/a_i]\} \end{aligned}$$

Here, $C[z/a_i]$ is a concept that is obtained from C by substituting every occurrence of the nominal schema $\{z\}$ in C with the nominal $\{a_i\}$. In addition, we need to impose the following restrictions to all tableau expansion rules other than grounding and Self-Ref-rule: they cannot be applied to the concept C if C is immediately ns-propagating, or directly connected, or has a nominal schema $\{x\}$ that occurs as a top-level conjunct/disjunct. This restriction ensures that no variable binding due to separate occurrences of nominal schemas is broken. Secondly, it prevents the propagation of concepts through the tableau structure which would make a nominal schema (or its negation) occur as a top-level conjunct/disjunct, i.e., such a propagation only occurs after the nominal schema is grounded. Note that a nominal schema (or its negation) may still occur already at the top-level of C_0, hence the above restriction makes sure that such a nominal schema is grounded before any other tableau expansion rule is applied.

Since we assume that no two axioms share the occurrences of the same nominal schema, each application of the grounding rule will only affect nominal schemas which originated from one axiom. Termination and correctness of the

algorithm can be shown by adapting the corresponding proofs for $\mathcal{SROIQ}$ [14] together with the following observation: any $\mathcal{SROIQV}$ knowledge base can always be completely (naively) grounded up front with the grounding rule to obtain a $\mathcal{SROIQ}$ knowledge base which can then be reasoned with using the standard $\mathcal{SROIQ}$ tableau algorithm. Moreover, taking into account the restrictions presented in the previous paragraph, a sequence of applications of tableau expansion rules with applications of the grounding rule interleaved in between can be simulated by a sequence of applications of tableau rule applications which is preceded by completely grounding the knowledge base up front using the grounding rule. In other words, the above restriction does not prevent the original tableau expansion rules to be applied at all, but rather, it only postpones such an application until after an appropriate grounding is done. Note that in the worst case, applications of the grounding rule will introduce exponentially many new concepts in the labels of tableau nodes. However, this grounding rule can still be incorporated without actually increasing the complexity of reasoning. In particular for $\mathcal{SROIQV}$, the optimal upper bound of complexity for knowledge base satisfiability is based on a reduction of theories of $\mathcal{C}^2$, the two-variable fragment of first-order logic with counting quantifiers [21]. For this setting, without spelling out the details, a variant of this grounding rule can actually be incorporated into the translation to $\mathcal{C}^2$ formulae. Hence, the following theorem holds:

Theorem 2. *Adding the grounding rule to reasoning algorithms for $\mathcal{SROIQ}$ yields a sound and complete algorithm to decide satisfiability of $\mathcal{SROIQV}$. Moreover, the grounding rule does not increase the complexity of reasoning.*

In fact, it can be shown that a similar result holds for reasoning in the tractable $\mathcal{SROEL}\mathcal{V}_n$, i.e., adding a variant of the grounding rule to existing reasoning algorithms for $\mathcal{SROEL}$ still retains tractability. Observe that the grounding rule as specified above is of non-deterministic nature because it non-deterministically chooses to which individual names should the grounding be done. It thus remains to devise a good heuristic in order to achieve a good performance in practical situations.

To demonstrate the advantage of having delayed grounding instead of grounding all nominal schemas up front, we consider again a knowledge base KB that contains the axiom (1) from page 3 together with some other axioms, as listed in Figure 1. Given the knowledge base KB, we ask whether KB entails the existence of a conflicting review assignment, i.e., whether the concept $\forall$hasConflictingAssignedPaper.$\bot$ is unsatisfiable w.r.t. KB.

The answer must be yes since a_1 and a_{1000} co-author p_0, but a_1 has review assignment on p_{999} whose authors include a_{1000}. This knowledge base has 2000 individual names. Hence, if we solve this entailment by grounding the first axiom up front, we would have 8×10^9 new axioms. On the other hand, with apropriate delayed groundings, namely grounding x to p_{999}, y to a_{1000} and z to ISWC, we could solve the entailment only in three grounding steps, plus a few more tableau expansion steps.

$$\exists \text{hasReviewAssignment}.((\{x\} \sqcap \exists \text{hasAuthor}.\{y\}) \sqcap (\{x\} \sqcap \exists \text{atVenue}.\{z\})) \\ \sqcap \exists \text{hasSubmittedPaper}.(\exists \text{hasAuthor}.\{y\} \sqcap \exists \text{atVenue}.\{z\}) \\ \sqsubseteq \exists \text{hasConflictingAssignedPaper}.\{x\}$$

$$\{p_0\} \sqsubseteq \exists \text{hasAuthor}.\{a_{1000}\} \sqcap \exists \text{hasAuthor}.\{a_1\}$$

$$\{p_i\} \sqsubseteq \exists \text{hasAuthor}.\{a_i\} \sqcap \exists \text{hasAuthor}.\{a_{i+1}\}$$

$$\{a_i\} \sqsubseteq \exists \text{hasSubmittedPaper}.\{p_{i-1}\} \sqcap \exists \text{hasSubmittedPaper}.\{p_i\}$$

$$\{a_{1000}\} \sqsubseteq \exists \text{hasSubmittedPaper}.\{p_{999}\} \sqcap \exists \text{hasSubmittedPaper}.\{p_0\}$$

$$\{p_j\} \sqsubseteq \exists \text{AtVenue}.\{\text{ISWC}\}$$

$$\{a_k\} \sqsubseteq \exists \text{hasReviewAssignment}.\{p_{k-4}\} \sqcap \exists \text{hasReviewAssignment}.\{p_{k-3}\}$$

$$\{a_1\} \sqsubseteq \exists \text{hasReviewAssignment}.\{p_{999}\} \sqcap \exists \text{hasReviewAssignment}.\{p_{998}\}$$

Fig. 1. Example for delayed grounding. $i = 1, \ldots, 999$, $j = 0, \ldots, 999$, $k = 4, \ldots, 1000$.

6 Adding Local Closure

We have seen in Theorem 1 that we can capture all of Datalog with nominal schemas. However, while Datalog is one of the most fundamental rules paradigms, there also exist numerous variants and extensions. Some of those considered most relevant for the Semantic Web are included or covered in the Rule Interchange Format RIF [15]. In particular, Horn logic is covered in the Basic Logic Dialect [3], and some of the key non-monotonic[22] rule systems are covered by the RIF Framework for Logic Dialects [4, Section 3.8].

So, while nominal schemas provide a tight integration of Datalog with OWL which has not been achieved before, it is fair to ask to what extent the approach allows to capture other rule paradigms. A comprehensive answer to this question is beyond the scope of this paper, but we can provide some evidence how a further reconciliation of the two paradigms might be achieved. We need to restrict this discussion to certain types of rule approaches, and thus will focus on extensions with non-montonic negation,[23] although we also see scope for other work.

Concerning non-monotonic extensions of Datalog, the primary current paradigm is based on the so-called *stable model semantics*[24] [8], which has given rise to *answer set programming* [23] as a major knowledge representation paradigm. This now begs the question how to extend $\mathcal{SROELV}$ (or other description logics

[22] Non-monotonicity refers to the fact that in such logics it is possible that the addition of new axioms causes the withdrawal of previously drawn conclusions. Non-monotonicity is closely related to the closed world assumption [9], and thus to the discussion of local closure for description logics around OWL.

[23] Since our approach is OWL-centric, i.e., decidability is important, undecidable rule languages such as RIF BLD [3] or logic programming with negation as failure and function symbols [11] are of limited interest in the context of this work.

[24] Note that a major alternative, the well-founded semantics [7], is closely related to the stable model semantics [11].

featuring nominal schemas) in such a way that the stable model semantics (or generalizations thereof) can be completely captured by introducing some form of local closed world modeling.[25] It would appear to be rather evident that this can be accomplished in some way, however a solution would be preferred which is intuitively appealing and easy to explain to users and developers.

We argue in [16, 30] that circumscription [24] provides such an intuitively appealing approach, and indeed we have developed a version of circumscription for description logics which overcomes major drawbacks of an earlier approach [5]; in particular, our modification features minimization (closure) of roles while staying decidable, while the approach from [5] is restricted to the closure of classes. In a nutshell, our approach—which we call *grounded circumscription*—makes it possible for the modeler to "close" an arbitrary choice of roles and classes, with the intuition that extensions of such classes or roles contain only those (pairs of) known individuals which are essentially required to be in those extensions. Another advantage of the circumscription approach is that it does not require the addition of any new syntax to the ontology language, as the closure information can be separated from the actual knowledge base, and the latter can be expressed as usual.

Now, in [6, 22] relationships between stable model semantics and circumscription have been worked out for logic programming (and thus for Datalog), and it was shown that the relation is rather tight. Grounded Circumscription over $\mathcal{SROELV}$ or $\mathcal{SROIQV}$ should thus subsume a non-monotonic rule paradigm which is closely related to the stable models approach. Spelling this out in detail, however, would be a different paper, and thus we defer this to future work.

7 Conclusion

We have presented a series of technical results and arguments, which support the claim that nominal schemas are a significant novel development in the quest to integrating OWL and rules. We consider this paper a stepping stone in further investigations into this quest. Naturally, a plethora of open issues remain to be investigated before a final conclusion can be reached.

- Efficient algorithms need to be developed, implemented and tested to show that nominal schemas can be reasoned with in a sufficiently efficient way.
- Further rule paradigms remain to be incorporated into the framework in order to achieve a further integration of diverse kinds of rule-based approaches into the OWL realm.
- Tool support for modeling and usage of nominal-schema-extended OWL remains to be developed.
- Application studies need to be undertaken in order to evaluate the usefulness, in practice, of the approach.

[25] See [16, 17] for a brief survey of existing approaches to local closed world extensions of description logics.

Acknowledgements. This work was supported by the National Science Foundation under award 1017225 "III: Small: TROn—Tractable Reasoning with Ontologies," and by State of Ohio Research Incentive funding in the Kno.e.CoM project. Adila Krisnadhi acknowledges support by a Fulbright Indonesia Presidential Scholarship PhD Grant 2010. We thank Markus Krötzsch for feedback on an early draft of this paper.

References

1. Beckett, D., Berners-Lee, T.: Turtle – Terse RDF Triple Language. W3C Team Submission 28 March 2011 (2011), available at http://www.w3.org/TeamSubmission/turtle/
2. Boley, H., Hallmark, G., Kifer, M., Paschke, A., Polleres, A., Reynolds, D. (eds.): RIF Core Dialect. W3C Recommendation 22 June 2010 (2010), available from http://www.w3.org/TR/rif-core/
3. Boley, H., Kifer, M. (eds.): RIF Basic Logic Dialect. W3C Recommendation 22 June 2010 (2010), available from http://www.w3.org/TR/rif-bld/
4. Boley, H., Kifer, M. (eds.): RIF Framework for Logic Dialects. W3C Recommendation 22 June 2010 (2010), available from http://www.w3.org/TR/rif-fld/
5. Bonatti, P.A., Lutz, C., Wolter, F.: The Complexity of Circumscription in Description Logic. Journal of Artificial Intelligence Research 35, 717–773 (2009)
6. Ferraris, P., Lee, J., Lifschitz, V.: Stable models and circumscription. Artificial Intelligence 175(1), 236–263 (2011)
7. van Gelder, A., Ross, K.A., Schlipf, J.S.: The well-founded semantics for general logic programs. Journal of the ACM 38(3), 620–650 (1991)
8. Gelfond, M., Lifschitz, V.: Classical Negation in Logic Programs and Disjunctive Databases. New Generation Computing 9, 365–385 (1991)
9. Grimm, S., Hitzler, P.: Semantic Matchmaking of Web Resources with Local Closed-World Reasoning. Int. J. of Electronic Commerce 12(2), 89–126 (2007)
10. Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P.F., Rudolph, S. (eds.): OWL 2 Web Ontology Language: Primer. W3C Recommendation 27 October 2009 (2009), available from http://www.w3.org/TR/owl2-primer/
11. Hitzler, P., Seda, A.K.: Mathematical Aspects of Logic Programming Semantics. CRC Press (2010)
12. Horridge, M., Bechhofer, S.: The OWL API: A Java API for OWL ontologies. Semantic Web 2(1), 11–21 (2011)
13. Horridge, M., Patel-Schneider, P. (eds.): OWL 2 Web Ontology Language: Manchester Syntax. W3C Recommendation (27 October 2009), available at http://www.w3.org/TR/owl2-manchester-syntax/
14. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible $\mathcal{SROIQ}$. In: Doherty, P., Mylopoulos, J., Welty, C. (eds.) Proc. 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'06). pp. 57–67. AAAI Press (2006)
15. Kifer, M., Boley, H. (eds.): RIF Overview. W3C Working Group Note 22 June 2010 (2010), available from http://www.w3.org/TR/rif-overview/
16. Krisnadhi, A., Sengupta, K., Hitzler, P.: Local closed world semantics: Keep it simple, stupid! In: Proceedings DL 2011 (2011), to appear
17. Krisnadhi, A., Maier, F., Hitzler, P.: OWL and Rules. In: Reasoning Web 2011. Lecture Notes in Computer Science, Springer, Heidelberg (2011), to appear

18. Krötzsch, M.: Efficient inferencing for OWL EL. In: Janhunen, T., Niemelä, I. (eds.) Proc. 12th European Conf. on Logics in Artificial Intelligence (JELIA'10). LNAI, vol. 6341, pp. 234–246. Springer (2010)
19. Krötzsch, M., Rudolph, S., Hitzler, P.: ELP: Tractable rules for OWL 2. In: Sheth, A., et al. (eds.) Proc. of the 7th International Semantic Web Conference (ISWC-08). Lecture Notes in Computer Science, vol. 5318, pp. 649–664. Springer (2008)
20. Krötzsch, M.: Description Logic Rules, Studies on the Semantic Web, vol. 008. IOS Press/AKA (2010)
21. Krötzsch, M., Maier, F., Krisnadhi, A.A., Hitzler, P.: A better uncle for OWL: Nominal schemas for integrating rules and ontologies. In: Sadagopan, S., Ramamritham, K., Kumar, A., Ravindra, M., Bertino, E., Kumar, R. (eds.) Proceedings of the 20th International World Wide Web Conference, WWW2011, Hyderabad, India, March/April 2011. pp. 645–654. ACM, New York (2011)
22. Lifschitz, V.: Twelve definitions of a stable model. In: de la Banda, M.G., Pontelli, E. (eds.) Logic Programming, 24th International Conference, ICLP 2008, Udine, Italy, December 9-13 2008, Proceedings. Lecture Notes in Computer Science, vol. 5366, pp. 37–51. Springer (2008)
23. Lifschitz, V.: What is answer set programming? In: Fox, D., Gomes, C.P. (eds.) Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008. pp. 1594–1597. AAAI Press (2008)
24. McCarthy, J.: Circumscription — a form of non-monotonic reasoning. Artificial Intelligence 13(1), 27–39 (1980)
25. Motik, B., Cuenca Grau, B., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C. (eds.): OWL 2 Web Ontology Language: Profiles. W3C Recommendation (27 October 2009), available at http://www.w3.org/TR/owl2-profiles/
26. Motik, B., Patel-Schneider, P., Parsia, B. (eds.): OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax. W3C Recommendation (27 October 2009), available at http://www.w3.org/TR/owl2-syntax/
27. Motik, B., Sattler, U., Studer, R.: Query answering for OWL-DL with rules. Journal of Web Semantics 3(1), 41–60 (2005)
28. OWL Working Group, W.: OWL 2 Web Ontology Language: Document Overview. W3C Recommendation (27 October 2009), available at http://www.w3.org/TR/owl2-overview/
29. Patel-Schneider, P., Motik, B. (eds.): OWL 2 Web Ontology Language: Mapping to RDF Graphs. W3C Recommendation (27 October 2009), available at http://www.w3.org/TR/owl2-mapping-to-rdf/
30. Sengupta, K., Krisnadhi, A., Hitzler, P.: Local closed world semantics: Grounded circumscription for OWL. Tech. rep., Kno.e.sis Center, Wright State University, Dayton, Ohio, USA (2011), submitted to ISWC2011. Available from http://www.pascal-hitzler.de/
31. Sirin, E., Parsia, B., Grau, B., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. Web Semantics: Science, Services and Agents on the World Wide Web 5(2), 51–53 (2007)

A Proof of Theorem 1

A proof of Theorem 1 is provided here. We use a transformation of rules into DL axioms that is slightly different than the one presented in Section 2—unary and binary predicates in rules are handled differently than higher arity predicates. This means that what is proven in Theorem 1 is actually slightly different. The reason for the modification is that the transformation here directly extends the one presented in [21], where rules with only unary and binary predicates are considered. The modified transformation is not problematic, in the sense that it is clear that the proof here can be readily altered to apply to rules translated as indicated in Section 2.

In the following, RB is a set of Datalog rules defined over a signature $\langle N_I, N_P, N_V \rangle$, where N_I is a set of constants, N_P is a set of n-ary predicates, and N_V is a set of variables. Each rule has the form $A_1, \ldots, A_n \rightarrow A$, where A and each A_i is of the form $p(t_1, \ldots, t_n)$, with $p \in N_P$ and each $t_i \in N_I \cup N_V$. We allow $\top$ and $\bot$ to occur in N_P, and they have their usual meaning. We will use $N_{P,i}$ ($N_{P,>i}$) to refer to the set of predicates of N_P with arity i (greater than i).

We briefly recount the semantics for Datalog. An *interpretation* $\mathcal{I}$ for RB consists of a domain of discourse $\Delta^{\mathcal{I}}$ together with a function $.^{\mathcal{I}}$ such that for each $a \in N_I$, $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$, and for each n-ary predicate $p \in N_P$, $p^{\mathcal{I}}$ is a subset of an n-ary relation over $\Delta^{\mathcal{I}}$. A *variable assignment* Z for $\mathcal{I}$ maps each variable $v \in N_V$ to some element of $\Delta^{\mathcal{I}}$: $Z(v) \in \Delta^{\mathcal{I}}$. Assignments must also satisfy the following restriction: if $Z(v) = d$, then there exists an $a \in N_I$ such that $a^{\mathcal{I}} = d$. This ensures that the rules of RB are *DL-safe*, i.e., that variables only refer to named individuals in the domain of discourse. For an arbitrary term in $N_I \cup N_V$, $t^{\mathcal{I},Z} = t^{\mathcal{I}}$ if t is a constant, and $t^{\mathcal{I},Z} = Z(t)$ if t is a variable. If $\mathcal{I}$ is an interpretation and Z an assignment, $\mathcal{I}$ and Z *satisfy* $p(t_1, \ldots, t_n)$, written $\mathcal{I}, Z \models p(t_1, \ldots, t_n)$, if and only if $(t_1^{\mathcal{I},Z}, \ldots, t_n^{\mathcal{I},Z}) \in p^{\mathcal{I}}$. For a set $\{A_1, \ldots, A_n\}$ of atomic formulas, $\mathcal{I}, Z \models \{A_1, \ldots, A_n\}$ if and only if $\mathcal{I}, Z \models A_i$ for each A_i. For a rule $B \rightarrow H$, $\mathcal{I}, Z \models B \rightarrow H$ if and only if $\mathcal{I}, Z \not\models B$ (that is, $\mathcal{I}$ and Z do not satisfy B) or else $\mathcal{I}, Z \models H$. Interpretation $\mathcal{I}$ is a *model* of formula X if and only if $\mathcal{I}, Z \models X$ for each assignment Z. A datalog program RB *entails* X, written $RB \models X$, if and only if each model of RB is a model of X.

RB can be embedded into an equisatisfiable $\mathcal{SROELV}$ knowledge base $\mathsf{dl}(RB)$ over signature $\langle N_I, N_C, N_R, N_V \rangle$, where N_C is a set of concept symbols and N_R is a set of role symbols. Here, $N_C = N_{P,1}$ and $N_R = N_{P,2} \cup \{U\} \cup S$, where U is the universal role and S is a special set of roles defined as follows: If $p \in N_{P,>2}$ has arity k, then $p_1, \ldots, p_k \in S$ are unique binary predicates associated with p; S is the set of all such predicates. The semantics for $\mathcal{SROELV}$ knowledge bases, which extends the usual description logic semantics with features to accommodate nominal schemas, is fully described in [21].

The rules for creating $\mathsf{dl}(RB)$ are given below. C and R refer to unary and binary predicates of RB, while p refers to a higher arity predicate.

1. $\mathsf{dl}(C(t)) := \exists U.(\{t\} \sqcap C)$;
2. $\mathsf{dl}(R(t,u)) := \exists U.(\{t\} \sqcap \exists R.\{u\})$;

3. $\mathsf{dl}(p(t_1,\ldots,t_k)) := \exists U.(\exists p_1.\{t_1\} \sqcap \ldots \sqcap \exists p_k.\{t_k\})$;
4. $\mathsf{dl}(A_1,\ldots,A_n \rightarrow H) := \mathsf{dl}(A_1) \sqcap \ldots \sqcap \mathsf{dl}(A_n) \sqsubseteq \mathsf{dl}(H)$;
5. $\mathsf{dl}(RB) := \{\mathsf{dl}(r) | r \in RB\}$.

Given an interpretation $\mathcal{I}$ for RB, we define a family $\mathrm{fam}(\mathcal{I})$ of interpretations based on $\mathcal{I}$. Without loss of generality, we assume that $\mathcal{I}$ is defined over a countably infinite domain of discourse. $\mathrm{fam}(\mathcal{I})$ is the set of all interpretations $\mathcal{J}$ which satisfy all of the following conditions.

1. $\Delta^{\mathcal{J}} := \Delta^{\mathcal{I}}$.
2. For each tuple $(d_1,\ldots,d_k) \in p^{\mathcal{I}}$, we assign an element e in $\Delta^{\mathcal{J}}$ to the tuple.
3. For each constant $a \in N_I$, $a^{\mathcal{J}} := a^{\mathcal{I}}$.
4. For each $C \in N_{P,1}$, $C^{\mathcal{J}} := C^{\mathcal{I}}$.
5. For each $R \in N_{P,2}$, $R^{\mathcal{J}} := R^{\mathcal{I}}$.
6. For each $p \in N_{P,>2}$, if $(d_1,\ldots,d_k) \in p^{\mathcal{I}}$, then $(e,d_i) \in p_i^{\mathcal{J}}$, where e is a domain element associated with $(d_1,\ldots,d_k)$ according to point 2 above.

Observe that any interpretation for $\mathsf{dl}(RB)$ can be reduced to one for RB: if $(e,d_1) \in p_1^{\mathcal{J}}$, ..., $(e,d_k) \in p_k^{\mathcal{J}}$, then $(d_1,\ldots,d_k) \in p^{\mathcal{I}}$. And so, for any interpretation $\mathcal{J}$ for $\mathsf{dl}(RB)$, there is an $\mathcal{I}$ for RB such that $\mathcal{J} \in \mathrm{fam}(\mathcal{I})$.

Below, since RB and $\mathsf{dl}(RB)$ share variables and since the interpretations described above are based on a common domain of discourse, we can assume that the variable assignments are the same for both RB and $\mathsf{dl}(RB)$. Also, we will often write Δ instead of, e.g., $\Delta^{\mathcal{I}}$.

Lemma 1. *Let A be an atom in RB, $\mathcal{I}$ an interpretation of RB, $\mathcal{J} \in fam(\mathcal{I})$, and Z an assignment.*
1. *$\mathcal{I}, Z \models A$ if and only if $\mathsf{dl}(A)^{\mathcal{J},Z} = \Delta$. and*
2. *$\mathcal{I}, Z \not\models A$ if and only if $\mathsf{dl}(A)^{\mathcal{J},Z} = \emptyset$.*

Proof. As noted in [21], since $\mathsf{dl}(A)$ has the form $\exists U.C$, $C^{\mathcal{J},Z} \neq \emptyset$ implies $\mathsf{dl}(A)^{\mathcal{J},Z} = \Delta$, and $C^{\mathcal{J},Z} = \emptyset$ implies $\mathsf{dl}(A)^{\mathcal{J},Z} = \emptyset$, and so $\mathsf{dl}(A)^{\mathcal{J},Z} \neq \emptyset$ if and only $\mathsf{dl}(A)^{\mathcal{J},Z} = \Delta$. Given this, it suffices to show $\mathsf{dl}(A)^{\mathcal{J},Z} \neq \emptyset$ *iff* $\mathcal{I}, Z \models A$.

Suppose $\mathcal{I}, Z \models A$ and let $d \in \Delta$. If $A = C(t)$, then $t^{\mathcal{J},Z} \in C^{\mathcal{J},Z}$, and so clearly $d \in (\exists U.(\{t\} \sqcap C))^{\mathcal{J},Z}$. Similarly, if $A = R(t,u)$, then $(t^{\mathcal{J},Z}, u^{\mathcal{J},Z}) \in R^{\mathcal{J}}$, and so clearly $d \in (\exists U.(\{t\} \sqcap \exists R.\{u\}))^{\mathcal{J},Z}$. If, instead, $A = p(t_1,\ldots,t_k)$, then $(t_1^{\mathcal{I},Z},\ldots,t_k^{\mathcal{I},Z}) \in p^{\mathcal{I}}$. By definition of $\mathcal{J}$, $(e,t_1^{\mathcal{J},Z}) \in p_1^{\mathcal{J}}$,..., $(e,t_k^{\mathcal{J},Z}) \in p_k^{\mathcal{J}}$ for some $e \in \Delta$. As such, $d \in (\exists U.(\exists p_1.\{t_1\} \sqcap \ldots \sqcap \exists p_k.\{t_k\}))^{\mathcal{J},Z}$.

Now suppose $d \in \mathsf{dl}(A)^{\mathcal{J},Z}$. If $\mathsf{dl}(A)$ is $\exists U.(\{t\} \sqcap C)$, then $(d, t^{\mathcal{J},Z}) \in U^{\mathcal{J},Z}$ and $t^{\mathcal{J},Z} \in C^{\mathcal{J},Z}$. From this, $t^{\mathcal{I},Z} \in C^{\mathcal{I},Z}$. If $\mathsf{dl}(A)$ is $\exists U.(\{t\} \sqcap \exists R.\{u\})^{\mathcal{J},Z}$, then $(t^{\mathcal{J},Z}, u^{\mathcal{J},Z}) \in R^{\mathcal{J},Z}$ and so $(t^{\mathcal{I},Z}, u^{\mathcal{I},Z}) \in R^{\mathcal{I},Z}$. If instead $\mathsf{dl}(A) = (\exists U.(\exists p_1.\{t_1\} \sqcap \ldots \sqcap \exists p_k.\{t_k\}))^{\mathcal{J},Z}$, then there is an $e \in \Delta^{\mathcal{I}}$ such that $(e,t_1^{\mathcal{J},Z}) \in p_1^{\mathcal{J},Z}$, ..., $(e,t_k^{\mathcal{J},Z}) \in p_k^{\mathcal{J},Z}$, and so $(t_1^{\mathcal{I},Z},\ldots,t_k^{\mathcal{I},Z}) \in p^{\mathcal{I},Z}$. In each case, $\mathcal{I}, Z \models A$. □

Lemma 2. *If $\mathcal{I}$ is an interpretation for RB and $\mathcal{J} \in fam(\mathcal{I})$, then $\mathcal{I}$ is a model of RB if and only if $\mathcal{J}$ is a model of $\mathsf{dl}(RB)$.*

Proof. Suppose $\mathcal{I}$ is a model of RB, and let Z be an assignment. Suppose $B \rightarrow H \in RB$. If $\mathcal{I}, Z \models H$, then $\mathsf{dl}(H)^{\mathcal{J},Z} = \Delta$ by Lemma 1. Similarly, if $\mathcal{I}, Z \not\models B$, then there exists a $B_i \in B$ such that $\mathcal{I}, Z \not\models B_i$. Again by Lemma 1, $\mathsf{dl}(B_i)^{\mathcal{J},Z} = \emptyset$, and so $\mathsf{dl}(B)^{\mathcal{J},Z} = \emptyset$. Either way, $\mathcal{J}, Z \models \mathsf{dl}(B \rightarrow H)$. Generalizing on Z and $B \rightarrow H$, $\mathcal{J}$ models $\mathsf{dl}(RB)$.

Now suppose $\mathcal{J}$ is a model of $\mathsf{dl}(RB)$. Suppose $B \rightarrow H \in RB$ and $\mathcal{I}, Z \models B$ for some assignment Z. For each $A \in B$, $\mathcal{I}, Z \models A$. By Lemma 1, $\mathsf{dl}(A)^{\mathcal{J},Z} = \Delta^{\mathcal{I}}$ for each $A \in B$, and so $\mathsf{dl}(B)^{\mathcal{J},Z} = \Delta^{\mathcal{I}}$. Since $\mathcal{J}, Z \models \mathsf{dl}(B \rightarrow H)$, it must be that $\mathsf{dl}(H)^{\mathcal{J},Z} = \Delta^{\mathcal{I}}$, and so $\mathcal{I}, Z \models H$ by Lemma 1. As such, $\mathcal{I}, Z \models B \rightarrow H$. Generalizing on Z and $B \rightarrow H$, $\mathcal{I}$ models RB. □

Theorem 1. Let RB be a set of rules and $\mathsf{dl}(RB)$ its $\mathcal{SROELV}$ translation. For any n-ary predicate p in RB and any n-tuple $(a_1, \ldots, a_k)$ of constants in RB, $RB \models p(a_1, \ldots, a_k)$ if and only if $\mathsf{dl}(RB) \models \top \sqsubseteq \exists U.(\exists p_1.\{a_1\} \sqcap \cdots \sqcap \exists p_n.\{a_n\})$.

Proof. Suppose $RB \models p(a_1, \ldots, a_k)$ and let $\mathcal{J}$ be a model of $\mathsf{dl}(RB)$ and Z a variable assignment. An interpretation $\mathcal{I}$ of RB can be constructed from $\mathcal{J}$. By Lemma 2, $\mathcal{I}$ is a model of RB and consequently of $p(a_1, \ldots, a_k)$. As such, $\mathcal{I}, Z \models p(a_1, \ldots, a_k)$. By Lemma 1, $\mathsf{dl}(p(a_1, \ldots, a_k))^{\mathcal{J},Z} = \Delta$. That is, $(\exists U.(\exists p_1.\{a_1\} \sqcap \ldots \sqcap \exists p_k.\{a_k\})^{\mathcal{J},Z} = \Delta$. From this, $\mathcal{J}, Z \models \top \sqsubseteq (\exists U.(\exists p_1.\{a_1\} \sqcap \ldots \sqcap \exists p_k.\{a_k\})$. Generalizing, $\mathcal{J}$ is a model of $\top \sqsubseteq (\exists U.(\exists p_1.\{a_1\} \sqcap \ldots \sqcap \exists p_k.\{a_k\})$.

Now suppose $\mathsf{dl}(KB) \models \top \sqsubseteq (\exists U.(\exists p_1.\{a_1\} \sqcap \ldots \sqcap \exists p_k.\{a_k\})$ and let $\mathcal{I}$ be a model of RB and Z a variable assignment. Let $\mathcal{J} \in \mathrm{fam}(\mathcal{I})$. By Lemma 2, $\mathcal{J}$ is a model of $\mathsf{dl}(RB)$ and consequently of $\mathcal{J}, Z \models \top \sqsubseteq (\exists U.(\exists p_1.\{a_1\} \sqcap \ldots \sqcap \exists p_k.\{a_k\})$. That is, $(\exists U.(\exists p_1.\{a_1\} \sqcap \ldots \sqcap \exists p_k.\{a_k\})^{\mathcal{J},Z} = \Delta$. By Lemma 1, $\mathcal{I}, Z \models p(a_1, \ldots, a_k)$. Generalizing on Z, $\mathcal{I}$ models $p(a_1, \ldots, a_k)$. □