8-2011

OWL and Rules

Adila Krisnadhi
Frederick Maier
Pascal Hitzler
pascal.hitzler@wright.edu

Follow this and additional works at: https://corescholar.libraries.wright.edu/cse
Part of the Computer Sciences Commons, and the Engineering Commons

Repository Citation
https://corescholar.libraries.wright.edu/cse/197

This Conference Proceeding is brought to you for free and open access by Wright State University's CORE Scholar. It has been accepted for inclusion in Computer Science and Engineering Faculty Publications by an authorized administrator of CORE Scholar. For more information, please contact corescholar@www.libraries.wright.edu, library-corescholar@wright.edu.
OWL and Rules

Adila Krisnadhi, Frederick Maier, and Pascal Hitzler
Kno.e.sis Center, Wright State University, Dayton, Ohio

Abstract. The relationship between the Web Ontology Language OWL and rule-based formalisms has been the subject of many discussions and research investigations, some of them controversial. From the many attempts to reconcile the two paradigms, we present some of the newest developments. More precisely, we show which kind of rules can be modeled in the current version of OWL, and we show how OWL can be extended to incorporate rules. We finally give references to a large body of work on rules and OWL.

1 Introduction

Since research into the Semantic Web began, there have been different paradigms for modeling ontologies. Two prominent approaches discussed at the very beginning are description logics [4] and rules, the latter in the wider sense of logic programming (e.g., in the form of F-Logic [41]). While both of these approaches are based on classical logic, they are sufficiently different that naive attempts to combine them were unsuccessful.

The Web Ontology Language OWL [33,61], which is now a W3C standard, was the primary DL-based formalism that resulted from these discussions [34,84]. Nevertheless, rule-based formalisms [76] proved successful, including in commercial applications, and they continued to be pursued after the development of OWL. This eventually led to the development of the W3C Recommendation RIF (Rule Interchange Format) [5,6].

The modeling split between description logics and rules has naturally led to a considerable number of efforts to understand the relationships between the two paradigms and to establish workable combinations of them. Some of the resulting formalisms and systems have proved to be successful (we give a partial list in Section 5). However, a formalism that successfully combines the two paradigms into a single ontology language—while at the same time remaining conceptually true to both of them and remaining computationally viable—has not been developed.

In this paper, we focus on new results in developing such a language. Specifically, we discuss adding what are called nominal schemas (first described in [51]) to description logics. The resulting language is entirely in the spirit of description logics (a point which we discuss in more detail in Section 4), and yet it allows basic rule patterns to be captured. This paper can be understood as a continuation of [30], in the sense that it discusses (in the same spirit) recent work on combining rules and ontologies.
After providing necessary terminology and technical preliminaries in Section 2, Sections 3 and 4 present material first described in [53] and [51], respectively. Specifically, Section 3 investigates the kinds of rules that are already expressible in the current OWL standard, and Section 4 shows how OWL can be extended to incorporate a significantly wider class of rules. In Section 5, we give pointers to other work combining rules and OWL. Section 6 concludes with some open issues for future research.

2 Preliminaries

For notation and terminology, and in particular for the definition of $SROIQ$, we follow the chapter by Sebastian Rudolph contained in this volume [84]. For a textbook introduction, see [34]; whereas for a comprehensive treatment of description logics, see [4]. We use description logic notation throughout. Recall that the description logic $SROIQ$ corresponds roughly to the OWL 2 DL profile of the Web Ontology Language [33,75]. Henceforth, by OWL we will understand OWL 2 DL. Some of the results discussed in this paper will also be closely related to the three tractable profiles of OWL 2 DL, namely OWL 2 EL, OWL 2 RL, and OWL 2 QL [64].

The description logic $SREL$, also known as $EL^+$, encompasses the following concept (class) and role (property) constructs:

- concept conjunction
- existential quantification
- Self
- role chains
- the universal role

The description logic $SROEL$ furthermore allows nominals. It essentially corresponds to OWL 2 EL [64]. The logic $SROIEL$ further allows the use of inverse roles.

Given a first-order logic signature, a Horn clause is a formula of the form $(\forall x_1)\cdots(\forall x_n)(B_1 \land \cdots \land B_k \rightarrow A)$, where each $x_i$ is a variable occurring in the formula and $A$ and each $B_i$ are atomic formulas, also called atoms. It is usual to omit the quantifiers and abbreviate the formula as

$$B_1 \land \cdots \land B_k \rightarrow A,$$

commonly known as a rule. Given such a rule, $A$ is called the head of the rule, while $B_1 \land \cdots \land B_k$ is called the body, and each $B_i$ is referred to as a body atom.

A function-free Horn clause is called a Datalog rule, and we will see many examples below. The Rule Interchange Format of the W3C [42] encompasses the RIF Core Dialect [5], which is essentially Datalog. In our discussion on

\footnote{Some additional role characteristics are usually also included, but this is not important for our discussion.}
integrating OWL and rules, we will mainly be concerned with Datalog using only unary and binary predicate symbols.

Semantically, we understand Datalog to be interpreted under the standard first-order predicate logic semantics. In some cases, we will refer to the Herbrand semantics, and will do so explicitly in each case.

3 Rules in OWL

In this section, we explore the question of which rules can be expressed in the current version of OWL. Results are adapted mainly from [53].

3.1 DLP and OWL 2 RL

It is rather obvious that certain DL axioms can be translated naively into rules:

\[ A \sqsubseteq B \text{ becomes } A(x) \rightarrow B(x) \]
\[ R \sqsubseteq S \text{ becomes } R(x, y) \rightarrow S(x, y) \]

DL axioms which involve only existential quantification and conjunction, and do so only on the left hand side of the concept inclusion, can also be translated easily:

\[ A \sqcap \exists R. \exists S. B \sqsubseteq C \text{ becomes } A(x) \land R(x, y) \land S(y, z) \land B(z) \rightarrow C(x) \]

However, for existential quantifiers on the right hand side of concept inclusion, there is no such translation.\(^2\)

Things become a bit trickier if we look at other DL concept constructors. Universal quantification occurring on the right hand side can be translated, but only when it is not on the left hand side.

\[ A \sqsubseteq \forall R. B \text{ becomes } A(x) \land R(x, y) \rightarrow B(y) \]

This is so because the axiom \[ A \sqsubseteq \forall R. B \] is equivalent to \[ \exists R^- . A \sqsubseteq B \]. Note, however, that the latter axiom requires an inverse role, whereas the former doesn’t.

Similarly, concept negation can be dealt with when occurring on the right hand side if it occurs together with disjunction, because an axiom like \[ A \sqsubseteq \neg B \sqcup C \] can be rewritten to \[ A \sqcap B \sqsubseteq C \], i.e.

\[ A \sqsubseteq \neg B \sqcup C \text{ becomes } A(x) \land B(x) \rightarrow C(x) \]

Cardinality restrictions can be translated as long as they can be rewritten, e.g., expressions such as \[ \geq 1 R.A \] would become \[ \exists R.A \], which can be handled if occurring on a left hand side. If we are allowed to use an equality symbol with the rules, then we can also express, e.g., functionality:

\[ \top \sqsubseteq \leq 1 R. \top \text{ becomes } R(x, y) \land R(x, z) \rightarrow y = z. \]

\(^2\) Unless we allow Skolemization which, however, does not result in a semantically equivalent expression, only in an equisatisfiable one.
Nominals can also be dealt with. They usually translate into the use of constants, and in some cases we also need equality:

\[ A \sqcap \exists R \{ b \} \sqsubseteq C \text{ becomes } A(x) \land R(x,b) \rightarrow C(x). \]
\[ \{ a \} \equiv \{ b \} \text{ becomes } \rightarrow a = b. \]

If we allow truth value predicates \( t \) and \( f \) on the rules side, then we can also express some axioms involving \( \top \) and \( \bot \):

\[ A \sqcap B \sqsubseteq \bot \text{ becomes } A(x) \land B(x) \rightarrow f. \]

Rules like the latter are usually called integrity constraints.

In some cases, DL axioms can be translated but result in more than one rule. This occurs, e.g., with disjunction on the left and with conjunction on the right hand side:

\[ A \sqsubseteq B \land C \text{ becomes } A(x) \rightarrow B(x) \text{ and } A(x) \rightarrow C(x) \]
\[ A \sqcup B \rightarrow C \text{ becomes } A(x) \rightarrow C(x) \text{ and } B(x) \rightarrow C(x) \]

If we look at this purely on the DL side, then the reason for this is that the first axiom indeed can be expressed as the two axioms \( A \sqsubseteq B \) and \( A \sqsubseteq C \), and likewise the second axiom can be expressed as the two axioms \( A \sqsubseteq C \) and \( B \sqsubseteq C \).

Armed with these observations, one is tempted to define a DL consisting only of axioms which can be translated into rules, e.g. as follows: A DL axiom \( \alpha \) can be translated into rules if, after translating \( \alpha \) into a first-order predicate logic expression \( \alpha' \), and after normalizing this expression into a set of clauses \( M \), each formula in \( M \) is a Horn clause (i.e., a rule). It needs to be noted, though, that this definition is dependent on the exact translation and normalization algorithm used: Is it allowed to use Skolemization? Is it allowed to use sophisticated algorithms which may, for example, eliminate tautological axioms which are not directly expressible as rules?\footnote{We could also consider the whole DL knowledge base as input to this process, and algorithms which do a sophisticated \textit{compilation} of the knowledge base. Indeed, such investigations have been carried out in a rather successful way, see e.g. \cite{62}, and also the notion of \textit{Horn DLs} resulting from this \cite{52}.}

If we stick to a naive translation and normalization,\footnote{It is difficult to exactly define “naive”—but essentially we mean a kind of direct translation of each axiom into equivalent rules, in the spirit of the examples we have given. How exactly the notion “naive” is understood, in fact, does not matter much for our discussion. See \cite{55} for a more conceptually inspired approach to defining rule fragments of DLs.} then the above observations are in fact the key idea behind the early language DLP \cite{28}, where the authors define a fragment of the DL \textit{SHOIQ} (and thus for the 2004 version of OWL \cite{61}) in this vein. DLP is discussed more in Section 5.3.
A naively adapted version of DLP, in fact, resulted in the OWL 2 profile OWL 2 RL [64]. In particular, in OWL 2 RL we can also deal with role chain axioms, which were not present in the 2004 version of OWL, and thus not part of the original DLP language:

\[ R \circ S \sqsubseteq T \text{ becomes } R(x, y) \land S(y, z) \rightarrow T(x, z) \]

However, the \texttt{Self} construct from OWL 2 DL did not make it into OWL 2 RL, although it in fact mediates another rather strong relationship to rules. We explore this in the following.

### 3.2 Rolification

Consider the sentence “All elephants are bigger than all mice.” [85], which is easily expressed by the rule

\[
\text{Elephant}(x) \land \text{Mouse}(y) \rightarrow \text{biggerThan}(x, y). \tag{1}
\]

It is indeed possible to translate this rule into OWL 2—however this involves a transformation which we call rolification.\(^6\) The rolification of a concept \(A\) is a (new) role \(R_A\) defined by the axiom \(A \equiv \exists R_A \text{Self} \). Armed with rolification, we can now express rule (1) by the axiom

\[
R_{\text{Elephant}} \circ U \circ R_{\text{Mouse}} \sqsubseteq \text{biggerThan},
\]

where \(U\) is the universal role, together with the two axioms for the rolifications of the concepts Elephant and Mouse,

\[
\text{Elephant} \equiv \exists R_{\text{Elephant}} \text{Self} \quad \text{and} \quad \text{Mouse} \equiv \exists R_{\text{Mouse}} \text{Self}.
\]

Note that this transformation is not exactly an equivalence transformation, since we introduce new role names. However, it is very akin to the technique of folding in logic programming, and the models of the rule stand in direct correspondence with the models of the resulting set of DL axioms, in the sense of a conservative extension\(^7\).

The rolification technique now makes it possible to translate further rules into DL syntax, in particular such rules where the rule head is a binary predicate:

\[
A(x) \land R(x, y) \rightarrow S(x, y) \text{ becomes } R_A \circ R \sqsubseteq S
\]

\[
A(y) \land R(x, y) \rightarrow S(x, y) \text{ becomes } R \circ R_A \sqsubseteq S
\]

\[
A(x) \land B(y) \land R(x, y) \rightarrow S(x, y) \text{ becomes } R_A \circ R \circ R_B \sqsubseteq S
\]

\(^5\) OWL 2 RL extends a naive adaptation of the DLP language by some additional features, such as keys, which are not relevant to our discussion.

\(^6\) It is also called man-man-ification, because one of the early examples involved a concept called \texttt{Man} [87].

\(^7\) That is, for every model \(I\) of the rule, there exists a model of the DL axioms which can be obtained from \(I\) by modifying the interpretation of the predicate symbols not appearing in the rule; in this case, the new roles \(R_{\text{Elephant}}\) and \(R_{\text{Mouse}}\). See [60] for further discussion about this definition.
A natural use of this form of axiom would be in specifying when a role restricts to a subrole, e.g., to state something like

\[
    \text{Woman}(x) \land \text{marriedTo}(x, y) \land \text{Man}(y) \rightarrow \text{hasHusband}(x, y),
\]
which translates to

\[
    R_{\text{Woman}} \circ \text{marriedTo} \circ R_{\text{Man}} \subseteq \text{hasHusband}
\]

However this has to be done with caution, because it would be natural for an axiom like

\[
    \text{hasHusband} \subseteq \text{marriedTo}
\]
to appear in the same knowledge base. This, however, is not allowed since it would violate regularity conditions on the RBox (see [84]).

To give another example for the rolification technique, consider the rule

\[
    \text{worksAt}(x, y) \land \text{University}(y) \land \text{supervises}(x, z) \land \text{PhDStudent}(z) \\
    \rightarrow \text{professorOf}(x, z),
\]
which can be expressed as

\[
    R_{\exists \text{worksAt. University}} \circ \text{supervises} \circ R_{\text{PhDStudent}} \subseteq \text{professorOf}.
\]

### 3.3 Description Logic Rules

Given the previous examples, it becomes natural to ask about sufficient conditions on rules for a possible translation into DL expressions using the rolification technique. Such conditions gave rise to the notion of Description Logic Rules (DL Rules) as introduced in [53]. The key intuition behind DL Rules is that bodies of such rules must be **tree-shaped** in a sense which we will now formally define. An example for a body which is not tree-shaped is \( R(x, y) \land S(y, z) \land T(x, z) \) — just consider each pair of variables connected by a role as an edge in a directed graph with the variables as vertices: for this example, the graph is not a tree, hence the body is not tree-shaped.

To formally define DL Rules, we have to fix the description logic. From our examples above we can see that the following expressive features are desirable: conjunction, existential quantification, role chains, Self, and the universal role. These are available in the polynomial-time DL \( \mathcal{SREL} \) (a.k.a. \( \mathcal{EL}^+ \)). To also deal with nominals, which are available in the polynomial DL \( \mathcal{SROEL} \) (a.k.a. \( \mathcal{EL}^{++} \)) which contains \( \mathcal{SREL} \) and is contained in OWL 2 EL. We have also seen above that inverse roles can be helpful, however they are not available in OWL 2 EL. They are available in \( \mathcal{SROI\mathcal{EL}} \) which is contained in OWL 2 DL.

Given a rule with body \( B \), we construct a directed graph as follows: First rename individuals (i.e., constants) such that each individual occurs only once — a body such as \( R(a, x) \land S(x, a) \) becomes \( R(a_1, x) \land S(x, a_2) \). Denote the resulting
new body by \( B' \). The vertices of the graph are then the variables and individuals occurring in \( B' \), and there is a directed edge between \( t \) and \( u \) if and only if there is an atom \( R(t, u) \) in \( B' \).

To illustrate this, consider the rule
\[
C(x) \land R(x, a) \land S(x, y) \land D(y) \land T(y, a) \rightarrow P(x, y).
\]
The resulting graph is 
\[
\text{\( a_1 \quad x \quad y \quad a_2 \).}
\]

**Definition 1.** We call a rule with head \( H \) tree-shaped (respectively, acyclic), if the following conditions hold.

- Each of the maximally connected components of the corresponding graph is in fact a tree (respectively, an acyclic graph)—or in other words, if it is a forest, i.e., a set of trees (respectively, a set of acyclic graphs).
- If \( H \) consists of an atom \( A(t) \) or \( R(t, u) \), then \( t \) is a root in the tree (respectively, in the acyclic graph).

To give some examples, the rule \( R(x, a) \land S(y, a) \rightarrow C(x) \) is tree-shaped, while the rule \( R(x, z) \land S(y, z) \rightarrow T(x, y) \) is acyclic but not tree-shaped. The first rule translates to \( R_{\exists R.\{a\}} \circ U \circ R_{\exists S.\{a\}} \subseteq R_C \) while the second translates to \( R \circ S^- \subseteq T \). Note the use of the inverse role in the second example, which cannot be avoided—this is typically the case for rules which are acyclic but not tree-shaped.

We now have the following results, which are slight adaptations from results in [53].

**Theorem 1.** The following hold.

- Every tree-shaped rule can be expressed in \( SROEL \).
- Every acyclic rule can be expressed in \( SROIEL \).

Description Logic Rules as defined in [50,53] now generalize Definition 1 by allowing unary predicates in rule atoms which are in fact concept expressions from the underlying DL. It is shown that, if this is done for \( SROIQ \) (resulting in \( SROIQ \) Rules), then there is a polynomial transformation of such rules back into \( SROIQ \). If it is done for \( SROEL \) or for OWL 2 RL, then the resulting language is polynomial. It is furthermore shown that \( SROEL \) can be captured completely by tree-shaped rules with the extension that rule heads may be of the form \( \exists R.A \), for a role \( R \) and an atomic concept \( A \).

A word of caution: Not every set of acyclic rules results in a set of axioms constituting a \( SROIQ \) knowledge base. This is due to the fact that not every set of \( SROIQ \) axioms is a \( SROIQ \) knowledge base: Restrictions on the use of non-simple roles must be adhered to, and the set of role chain axioms must be regular (see [84]).

We close this part with a rule that is not acyclic:
\[
\text{hasReviewAssignment}(v, x) \land \text{hasAuthor}(x, y) \land \text{atVenue}(x, z) \\
\land \text{hasSubmittedPaper}(v, u) \land \text{hasAuthor}(u, y) \land \text{atVenue}(u, z) \quad (2) \\
\rightarrow \text{hasConflictingAssignedPaper}(v, x)
\]
The corresponding graph is the following.

\[ \xymatrix{ x \ar[r] & y \ar[dl] & z \ar[dl] \ar[l] \cr v \ar[u] & \ar[l] \ar[r] & u } \]

Note, however, that if \( y \) and \( z \) were constants, then the rule would be tree-shaped and could be expressed in \( SROEL \) as

\[
R \exists \text{hasSubmittedPaper} . (\exists \text{hasAuthor} . \{ y \} \sqcap \exists \text{atVenue} . \{ z \}) \circ \exists \text{hasReviewAssignment} \\
\circ R \exists \text{hasAuthor} . \{ y \} \sqcap \exists \text{atVenue} . \{ z \} \\
\sqsubseteq \exists \text{hasConflictingAssignedPaper}.
\]

4 Rules Plus OWL

Theorem 1 allows us to identify rules expressible as DL axioms in a rather natural way. This, however, is only one step towards reconciling the rule-based and DL-based paradigms, as there are clearly additional (and desirable) things that are expressible in rules but which do not fit the format of Theorem 1. In this section, we discuss using nominal schemas \[51\] to significantly widen the class of rules expressible in a DL language. We believe nominal schemas provide one of the more seamless methods of integrating rule-based and DL-based ontology languages to date. But before we arrive at that, we will provide some relevant historical background.

4.1 DL-safe Rules, DL-safe Variables and ELP

Although DLs and rule languages are decidable fragments of first-order logic, it is well known that an unrestricted combination of both leads to undecidability. Intuitively, this is because many DLs rely on the so-called tree model property to retain decidability, and this property is lost when rules come into play \[74\].

Another related source of problems, which may similarly lead to undecidability or complexity blow-up, is the fact that DL knowledge bases typically entail the existence of anonymous individuals within a possibly infinite domain. This makes things difficult in the presence of rules, which generally apply to all individuals in the domain \[54\]. Therefore, a crucial step when one wants to combine the rule-based paradigm and the DL-based paradigm in one ontology is to come up

\footnote{A DL is said to have the \textit{tree model property} when every satisfiable formula in it has a model which is of a tree-shape, where tree-shapedness is understood in a similar way as discussed in Section 3.3. Note that there are decidable DLs in which this property is not satisfied. In such DLs, decidability can be recovered by applying sophisticated strategies in the reasoning algorithm, e.g., blocking, see \[4\].}
with some safety criterion to ensure decidability or certain complexity bounds for reasoning over the combined language.

A prominent example of such a safety criterion is the notion of DL-safe rules [74] (see Sections 5.1 and 5.2). These restrict the applicability of rules in the combined knowledge base to named individuals, i.e., to individuals explicitly mentioned in the knowledge base. This guarantees decidability because there can only be a finite number of named individuals in the knowledge base.

More relevant to the current discussion is that DL-safe rules can be added to SROEL without losing tractability, under the restriction that there is a global bound on the number of variables which can occur in each rule. The resulting language, called ELP, is a tractable ontology language based on the DL rules framework (discussed in Section 3.3) that generalizes DL-safe rules by building this safety criteria directly into the semantics of variables [54]. Syntactically, an ELP rule base is a set of rules with function-free, unary and binary atoms whose predicate symbols are formed from SROEL concept and role expressions.

We assume, in the signature of ELP, that the set of individuals is finite and contains only those named individuals occurring in the knowledge base. In addition, the DL-safety criteria is built into the semantics of variables as follows: from the set of variables that is a part of ELP’s signature, we specify a fixed subset that contains precisely those variables which can only be assigned to named individuals. Let us revisit the following example (2) from page 7:

\[
\text{hasReviewAssignment}(v, x) \land \text{hasAuthor}(x, y) \land \text{atVenue}(x, z) \\
\land \text{hasSubmittedPaper}(v, u) \land \text{hasAuthor}(u, y) \land \text{atVenue}(u, z) \\
\rightarrow \text{hasConflictingAssignedPaper}(v, x)
\]

This rule is in ELP if the variables \(y\) and \(z\) are DL-safe variables. The intuition behind DL-safe variables is so that we can regain a tree-shape for the rule when these safe variables are replaced with named individuals from the knowledge base.

The tree-shapedness notion for ELP rules is based on Definition 1 with the following exceptions:

- there can be more than one tree edge (must be of the same direction) between two vertices; this corresponds to role conjunctions; if there is more than one tree edge between two vertices, those edges must correspond to simple roles only;
- atoms of the form \(R(x, x)\) are ignored when defining a path in the tree, i.e., local reflexivity is allowed; \(R\) must be simple.

A rule base in ELP contains those rules whose atoms use SROEL concepts and role expressions and satisfy the tree-shapedness notion above, and which may in addition contain rules of the form \(R(x, y) \rightarrow C(y)\) that satisfy: for each such rule, if the rule base contains a rule \(B \rightarrow H\) with \(R(t, z) \in H\), then \(C(z) \in B\).

The following theorem from [54] gives the tractability result for ELP.

**Theorem 2.** Satisfiability of any ELP rule base can be decided in time polynomial in the size of the rule base.
The above result from ELP is an important milestone in the effort to reconcile DL-based and rule-based paradigms in ontology languages. Not only because of the tractability of reasoning, but also because of the fact that it subsumes both SROEL (i.e., OWL 2 EL) and DLP (i.e., most of OWL 2 RL) in the following sense [54].

Theorem 3. Given any ground atom \( \alpha \) of the form \( C(a) \) or \( R(a,b) \), a DLP rule base \( \mathcal{R} \), and a SROEL knowledge base \( \mathcal{K} \), there exists an ELP rule base \( \mathcal{R}' \) such that if \( \mathcal{R} \models \alpha \) or \( \mathcal{K} \models \alpha \) then \( \mathcal{R}' \models \alpha \), and if \( \mathcal{R}' \models \alpha \) then \( \mathcal{R} \cup \mathcal{K} \models \alpha \), and \( \mathcal{R}' \) can be computed in linear time.

In fact, the expressivity of ELP exceeds that of SROEL because it admits conjunctions of simple roles and limited range restrictions (expressed using rules). Note however, that ELP is clearly still a hybrid language because it uses both rule-based and DL-based syntax. This hybrid nature of ELP makes it rather complicated to integrate with OWL 2 DL standard which is roughly based on the DL paradigm. This becomes one of the motivations for the development of nominal schemas which is discussed in the sequel.

4.2 Nominal Schemas: Intuitive Idea

The notion of DL-safe variables in the previous section gives an insight on how to integrate rule-based and DL-based paradigms in a DL framework and how such integration can then be adapted quite easily into the current OWL syntax. The key observation is obtained from the fact that a DL-safe variable essentially represents all possible groundings to named individuals in the knowledge base. What we need is a way to specify this explicitly within DL syntax. This was realized in a new DL construct called nominal schemas, which syntactically resemble nominals [51]. In this paper, we consider the following DL languages:

- SROIQ\(V(\mathcal{B}, \times)\) that is an extension of SROIQ (which roughly corresponds to OWL 2 DL) with Boolean operators on roles, concept products, and nominal schemas;
- SROEL\(V(\top, \times)\) that is an extension of SROEL (which roughly corresponds to OWL 2 EL) with role conjunction, concept products and nominal schemas. For the latter, we will mainly speak about the tractable fragments SROEL\(V(\top, \times), n \geq 0\), which can be obtained from SROEL\(V(\top, \times)\) by restricting the number of occurrences of certain nominal schemas that will be introduced later.

To understand why nominal schemas allow a seamless integration of rules within DL-based syntax, note that in ELP, variables can essentially be categorized into two types: DL-safe variables which must be bound only to named individuals, and non-DL-safe variables which may represent anonymous individuals in the domain of the knowledge base. Thus, if we want to use a DL-based syntax, we can just hide the anonymous individuals inside the concept and role expressions and then deal with DL-safe variables separately. This is where nominal schemas are used.
One characteristic feature of rules that is brought into DL axioms by nominal schemas is variable bindings. Consider the following rule

$$\text{hasChild}(x, y) \land \text{hasChild}(x, z) \land \text{classmate}(y, z) \rightarrow C(x)$$

which defines a concept $C$ of parents with at least children which are classmates (consider the role classmate to be irreflexive). This rule is not tree-shaped as it induces two paths from $x$ to $z$. Moreover, the variable $z$ which occurs in different atoms must be bound to the same individual. This cannot be simulated in DLs unless we are equipped with nominal schemas as follows:

$$\exists \text{hasChild}.\{z\} \sqcap \exists \text{hasChild}.\exists \text{classmate}.\{z\} \sqsubseteq C$$

The following example—see (2) on page 7 and (3) on page 9—is expressed in SROELV$_\cap$($\sqcap$, $\times$). It states that somebody has a conflicting review assignment (paper $x$) if this person has a paper submitted at the same event which is co-authored by one of the authors of paper $x$.

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

$$\sqcap \exists \text{hasSubmittedPaper}.(\exists \text{hasAuthor}.\{y\} \sqcap \exists \text{atVenue}.\{z\})$$

$$\sqsubseteq \exists \text{hasConflictingAssignedPaper}.\{x\}$$

(4)

The last example does not induce tree-shaped structures, the fact of which is quite clear if we rewrite it as a rule. There, the tree-shaped structure can be recovered when $x$ is ground as a named individual. This particular insight is exploited to show the tractability of reasoning for SROELV$_n$($\cap$, $\times$).

Formally, this is done by introducing the notion of safe environment.\footnote{Definition 2 is slightly more general than the one presented in [51], leading to a slightly more general polynomial language.}

**Definition 2.** An occurrence of nominal schema $\{x\}$ in a concept $C$ is safe if $C$ contains a sub-concept of the form $\{v\} \sqcap \exists R.D$ for some nominal schema or nominal $\{v\}$ such that $\{x\}$ is the only nominal schema that occurs (possibly more than once) in $D$. In this case, $\{v\} \sqcap \exists R.D$ is a safe environment for this occurrence of $\{x\}$, sometimes written as $S(v, x)$.

The virtue of safe environments lies in the fact that, algorithmically, safe occurrences of nominal schemas can essentially be handled separately from the axiom in which they occur, thus avoiding a combinatorial explosion through grounding, provided that there is a global bound on the number of occurrences of those safe nominal schemas in each axiom [51]—we will return to this issue in the proof sketch, and subsequent examples, of Theorem 5 below. The following definition captures this idea, and it will be explained in more detail further below.

**Definition 3.** Let $n \geq 0$ be an integer. A SROELV($\sqcap$, $\times$) knowledge base $KB$ is a SROELV$_n$($\sqcap$, $\times$) knowledge base if in each of its axioms $C \sqsubseteq D$, there are at most $n$ nominal schemas appearing more than once in non-safe form, and all remaining nominal schemas appear only in $C$. 
Note the dependency of the definition on the positive integer \( n \), which is a global bound on the number of nominal schemas which can occur (more than once in non-safe form) in any axiom. Without this global bound we would not be able to retain tractability of reasoning.

Returning to our example axiom (4) above, we see that it indeed lies in \( SROELV_1(\sqcap, \times) \).

### 4.3 Nominal Schemas: Formal Definitions and Results

We now formally introduce syntax and semantics of nominal schema. As indicated in section 4.2, we introduce two new languages: \( SROIQV(B_s, \times) \) and \( SROELV_n(\sqcap, \times) \). We will start with the former and then introduce the latter as its sublanguage. Let the set of individual names \( N_I \), the set of concept names \( N_C \), and the set of role names \( N_R \) form the signature of the DL \( SROIQ \) as defined in [84]. The signature of \( SROIQV(B_s, \times) \) is then formed from \( N_I, N_C, N_R \), and additionally the set of variables \( N_V \). We also assume that these sets are finite and pairwise disjoint. As already seen from the earlier examples, we use lower case letters \( x, y, z, \ldots \) to denote variables. Furthermore, the set of role names \( N_R \) is partitioned into disjoint sets \( N_{R_s} \) of simple role names and \( N_{R_n} \) of non-simple role names. Note that this partition is fixed from the signature, i.e., is not defined based on syntactic properties, e.g., how it occurs in the TBox or ABox, etc. This simplifies the presentation.

The set of \( SROIQV(B_s, \times) \) roles \( R \) is the union of two (non-disjoint) sets: the set of simple roles \( R^s \) and the set of non-simple roles \( R^n \) where \( R^s \) consists of (defined inductively):

- all simple role names;
- inverses of simple role names, i.e., \( R^{-} \) for every simple role name \( R \);
- the universal role \( U \);
- \( \neg R, R \sqcap S \) and \( R \sqcup S \) where \( R, S \) are simple roles in \( R^s \);
- the concept products \( A \times B \) where \( A, B \) are concept names;

and \( R^n \) consists of (defined inductively):

- all non-simple role names;
- inverses of non-simple role names, i.e., \( R^{-} \) for every non-simple role name \( R \);
- the universal role \( U \);
- the concept products \( A \times B \) where \( A, B \) are concept names.

The set of \( SROIQV(B_s, \times) \) concepts \( C \) consists of (defined inductively):

- the top concept \( \top \) and the bottom concept \( \bot \);
- every concept name \( A \in N_C \);
- \( \{a\} \) for every individual name \( a \in N_I \);
- \( \{v\} \) for every variable \( v \in N_V \);
- \( \neg C, C \sqcap D \) and \( C \sqcup D \) where \( C, D \) are concepts;
- \( \exists R.C \) and \( \forall R.C \) where \( R \) is a role;
- \( \exists R. \text{Self}, \leq kR.C \) and \( \geq kR.C \) where \( R \) is a simple role, \( k \) any non-negative integer and \( C \) concept.

Concepts \( \{a\} \) with \( a \in N_I \) are called nominals and concepts \( \{v\} \) with \( v \in N_V \) are called nominal schemas. Essentially, concepts and roles for \( SROIQV(B_s, \times) \) are \( SROIQ \) concepts and roles extended with concept product (indicated with \( \times \)), nominal schema (indicated with the letter \( V \)) and Boolean role constructors (indicated with the letter \( B_s \)).

A \( SROIQV(B_s, \times) \) knowledge base consist of RBox, TBox and ABox axioms with syntax defined as usual. The regularity condition for \( SROIQ \) knowledge bases also applies for \( SROIQV(B_s, \times) \) knowledge bases.

The semantics of \( SROIQV(B_s, \times) \), like that of \( SROIQ \), is based on interpretations \( I = (\Delta^I, \tau^I) \) with \( \Delta^I \) the domain of \( I \) and \( \tau^I \) the interpretation mapping. But we need an additional component for interpretation of variables. This is realized by associating a variable assignment \( Z : N_V \rightarrow \Delta^I \) for the interpretation \( I \). The assignment \( Z \) is such that for each \( v \in N_V \), \( Z(v) = a^I \) for some \( a \in N_I \). Another interpretation mapping \( \tau^I, Z \) is then defined that reflects both \( I \) and \( Z \). The base definition of \( \tau^I, Z \) starts from concept names, role names, individual names and variables as follows:

\[
\begin{align*}
A^I, Z &= A^I \subseteq \Delta^I & R^I, Z &= R^I \subseteq \Delta^I \times \Delta^I \\
a^I, Z &= a^I \in \Delta^I & x^I, Z &= Z(x) \in \Delta^I
\end{align*}
\]

Extending \( \tau^I, Z \) for complex concepts and roles is straightforward and very similar to the way \( \tau^I \) is extended to them in \( SROIQ \). The following are for complex concepts:

\[
\begin{align*}
\top^I, Z &= \Delta^I & \bot^I, Z &= \emptyset & \{t\}^I, Z &= \{t^I, Z\} \text{ for } t \in N_I \cup N_V \\
(\exists R.C)^I, Z &= \{\delta \mid \text{there is } \epsilon \text{ with } \langle \delta, \epsilon \rangle \in R^I, Z \text{ and } \epsilon \in C^I, Z\} \\
(\forall R.C)^I, Z &= \{\delta \mid \text{for all } \epsilon \text{ with } \langle \delta, \epsilon \rangle \in R^I, Z, \text{ we have } \epsilon \in C^I, Z\} \\
(\exists R. \text{Self})^I, Z &= \{\delta \mid \langle \delta, \delta \rangle \in R^I, Z\} \\
(\neg C)^I, Z &= \Delta^I \setminus C^I, Z \\
(C \cap D)^I, Z &= C^I, Z \cap D^I, Z & (C \cup D)^I, Z &= C^I, Z \cup D^I, Z \\
(\leq kR.C)^I, Z &= \{\delta \mid \#\{\delta, \epsilon \} \in R^I, Z \mid \epsilon \in C^I, Z\} \leq k\} \\
(\geq kR.C)^I, Z &= \{\delta \mid \#\{\delta, \epsilon \} \in R^I, Z \mid \epsilon \in C^I, Z\} \geq k\}
\end{align*}
\]

For roles, the following holds:

\[
\begin{align*}
U^I, Z &= \Delta^I \times \Delta^I \\
(R^{-})^I, Z &= \{\langle \delta, \epsilon \rangle \mid \epsilon \delta \in R^I, Z\} \\
(A \times B)^I, Z &= \{\langle \delta, \epsilon \rangle \mid \delta \in A^I, Z \text{ and } \epsilon \in B^I, Z\} \\
(\neg R)^I, Z &= (\Delta^I \times \Delta^I) \setminus R^I, Z \\
(R \cap S)^I, Z &= R^I, Z \cap S^I, Z & (R \cup S)^I, Z &= R^I, Z \cup S^I, Z
\end{align*}
\]
Let $\mathcal{I}$ be an interpretation and $Z$ a variable assignment for $\mathcal{I}$. For a $\text{SROIQV}(B_s, \times)$ axiom $\alpha$, we say, $\mathcal{I}$ and $Z$ satisfy $\alpha$ (written $\mathcal{I}, Z \models \alpha$) if the following holds for the corresponding form of $\alpha$:

- $\mathcal{I}, Z \models A(t)$ iff $t^{\mathcal{I}, Z} \in A^{\mathcal{I}, Z}$
- $\mathcal{I}, Z \models R(t, u)$ iff $(t^{\mathcal{I}, Z}, u^{\mathcal{I}, Z}) \in R^{\mathcal{I}, Z}$
- $\mathcal{I}, Z \models C \sqsubseteq D$ iff $C^{\mathcal{I}, Z} \subseteq D^{\mathcal{I}, Z}$
- $\mathcal{I}, Z \models R \sqsubseteq S$ iff $R^{\mathcal{I}, Z} \subseteq S^{\mathcal{I}, Z}$

where ‘$\circ$’ denotes the usual composition of binary relations.

$I$ satisfies $\alpha$, written $\mathcal{I} \models \alpha$, if $\mathcal{I}, Z \models \alpha$ for every variable assignment $Z$ for $\mathcal{I}$. $\mathcal{I}$ satisfies a $\text{SROIQV}(B_s, \times)$ knowledge base $KB$, written $\mathcal{I} \models KB$, if $\mathcal{I} \models \alpha$ for every $\alpha \in KB$. In this case, we say $KB$ is satisfiable (has a model). $KB$ entails an axiom $\alpha$, written $KB \models \alpha$, if all models of $KB$ are also models of $\alpha$.

It is known that reasoning in $\text{SROIQ}(B_s)$ is $\text{N2ExpTime}$-complete — thus, of the same complexity as $\text{SROIQ}$ — where this logic is an extension of $\text{SROIQ}$ with Boolean role operators (and concept products too, since concept products can be simulated using role negations) [83]. Reasoning in $\text{SROIQV}(B_s, \times)$ can thus be done by grounding the nominal schemas first, i.e., substituting each nominal schema with finitely many named individuals it may represent, resulting in a knowledge base in $\text{SROIQ}(B_s)$, and then proceed with the reasoning algorithm for $\text{SROIQ}(B_s)$. If each axiom contains $m$ different nominal schemas, and there are a total of $n$ axioms in the knowledge base, then this naive grounding will generate $n \cdot |N_i|^m$ new axioms, i.e., a number exponential in the size of the input knowledge base if there is no global bound on $m$. However, as stated in the following theorem, adding nominal schema does not actually increase the complexity [51].

**Theorem 4.** The problem of deciding satisfiability of a $\text{SROIQV}(B_s, \times)$ knowledge base is $\text{N2ExpTime}$-complete.

Another problem of obvious interest is to identify a fragment of the language $\text{SROIQV}(B_s, \times)$ that admits nominal schemas as one of its constructors, but is still tractable in reasoning.

As mentioned in Section 4.2, the idea of nominal schemas is inspired from the use of DL-safe variables in $\text{ELP}$ which is a tractable extension of $\text{SROEL}$. So, obvious candidates to look at are extensions of $\text{SROEL}$ with nominal schemas. In [51], the DLs $\text{SROEL}_n(\cap, \times)$ were presented as such candidates. These DLs are extensions of $\text{SROEL}(\cap, \times)$ which are defined for each integer $n \geq 0$. The number $n$ that is a part of the language definition provides a global bound that restricts the number of “unsafe” occurrences of nominal schemas in an axiom.

Recall that occurrences of nominal schemas in an axiom provides variable bindings which are a characteristic feature of rules, but not of DL axioms. In general, such bindings may represent complex dependencies that are difficult to
simplify. The naive way to process nominal schemas is by grounding them all to every possible replacement with named individuals in the knowledge base. This obviously leads to intractability as this naive grounding introduces exponential blow-up in the size of the knowledge base.

To achieve tractability, a better reduction on the number of nominal schemas is needed. Fortunately, by borrowing insight from ELP, we understood that there are special cases in which nominal schemas on the left-hand side of TBox axioms can be eliminated or separated using independent axioms. The idea from ELP is that when the dependencies expressed in a rule body are tree-shaped, the rule can be reduced to a small set of normalized rules, each of which contains a limited number of variables. This idea was then exploited to obtain the tractability results of ELP [54].

Elevating this idea to SROELV, we view variables in rules as either “hidden” in the concept expression or as occurring explicitly as nominal schemas. Note that in [54], tree-shapedness only refers to variables and not constants which correspond to nominals in our case here. Thus, nominals can be used to disconnect a dependency structure in a concept. For example, consider the concept

\[ A \sqcap \exists R.\{z\} \sqcap \exists S.(B \sqcap \exists T.\{z\}) \]

which corresponds to the rule body

\[ A(x) \land R(x, z) \land S(x, y) \land B(y) \land T(y, z). \]

The tree-shapedness of the rule is recovered when \( y \) is actually a constant. In the corresponding concept, this means a nominal in the place of the concept \( B \). When this is the case, the nominal schema \( \{z\} \) within the last conjunct of the example concept occurs in a safe environment, which is the safety criteria that we need. The formal Definition 2 generalizes this to the case where \( y \), as in the example above, is a nominal schema instead of a nominal.

We now give a formal definition of the DL SROELV\((\sqcap, \times)\) — and thus, of SROELV\(_n\)(\(\sqcap, \times\)) for every \( n \geq 0 \). We define a SROELV\((\sqcap, \times)\) concept as a SROIQV\((B_s, \times)\) concept that may contain \( \top, \bot \), conjunctions, existential restrictions, self restrictions, nominals and nominal schemas, but that does not contain disjunctions, negations, universal restrictions, and number restrictions. A SROELV\((\sqcap, \times)\) role is a SROIQV\((B_s, \times)\) role (simple or non-simple) which may contain role conjunction (for simple roles) and the universal role, but no inverse roles, role disjunction or role negation. TBox, RBox and ABox axioms for SROELV\((\sqcap, \times)\) are TBox, RBox, and ABox axioms in SROIQV\((B_s, \times)\) that use only SROELV\((\sqcap, \times)\) concepts and roles. Furthermore, every SROELV\((\sqcap, \times)\) knowledge base satisfies the following restriction.

**Definition 4.** Let \( KB \) be a knowledge base and \( R \) a role name. Let \( \text{ran}(R) \) be the set of all concept names \( B \) for which there is a set \( \{R \sqsubseteq R_1, R_1 \sqsubseteq R_2, \ldots, R_{n-1} \sqsubseteq R_n, R_n \sqsubseteq A \times B\} \sqsubseteq KB \) with \( n > 0 \) and \( R_0 = R \). We impose that every SROELV\((\sqcap, \times)\) knowledge base must satisfy admissibility range restrictions for every role inclusion axiom in it as follows: \( R_1 \circ \ldots \circ R_n \sqsubseteq S \) implies \( \text{ran}(S) \sqsubseteq \text{ran}(R_n) \) and \( R_1 \sqcap R_2 \sqsubseteq S \) implies \( \text{ran}(S) \sqsubseteq \text{ran}(R_1) \cup \text{ran}(R_2) \).
This admissibility criteria is from $SROEL(\sqcap, \times)$, as defined in [49].

Finally, $SROELV_n(\sqcap, \times)$ concepts and roles are $SROEL(\sqcap, \times)$ concepts and roles. Also, $SROELV_n(\sqcap, \times)$ knowledge bases are $SROEL(\sqcap, \times)$ knowledge bases that satisfies Definition 3. For $SROELV_n(\sqcap, \times)$, we have obtain the following result for every integer $n \geq 0$.

**Theorem 5.** If $KB$ is a $SROELV_n(\sqcap, \times)$ knowledge base of size $s$, satisfiability of $KB$ can be decided in time proportional to $s^n$. If $n$ is constant, then the problem is P-complete.

A full proof of this theorem can be found in [51]. We explain the key idea of the proof by means of our running example (4). Note that a naive grounding, as explained above, would result in $|N_I|$ new axioms (without nominal schemas, but with nominals). To decrease this figure without loss of completeness or soundness, we take advantage of safe environments—the rationale behind this being that safe environments can be handled separately from the rest of the axiom, as follows.\(^{10}\)

We first replace, in the axiom, the safe environments by a single nominal, and we do this replacement for every nominal in the knowledge base. That is, we obtain $|N_I|$ new axioms as follows, where $a_i$ ranges over all elements of $N_I$. Note that we also replaced the remaining occurrence of the nominal schema $\{x\}$ accordingly.\(^{11}\)

\[
\exists \text{hasReviewAssignment}.\{(a_i) \cap \{a_i\}\}
\]

\[
\sqcap \exists \text{hasSubmittedPaper}.(\exists \text{hasAuthor}.\{y\} \sqcap \exists \text{atVenue}.\{z\})
\]

\[
\sqsubseteq \exists \text{hasConflictingAssignedPaper}.\{a_i\}
\]

Next, we replace the remaining occurrences of $\{y\}$ and $\{z\}$ (note that there can be at most one for each of these nominal schemas, per definition of the language $SROELV_n(\sqcap, \times)$) by new concept names $O_y$ and $O_z$ (when subsequently converting other axioms, new concept names need to be used).

\[
\exists \text{hasReviewAssignment}.\{(a_i) \cap \{a_i\}\}
\]

\[
\sqcap \exists \text{hasSubmittedPaper}.(\exists \text{hasAuthor}.O_y \sqcap \exists \text{atVenue}.O_z)
\]

\[
\sqsubseteq \exists \text{hasConflictingAssignedPaper}.\{a_i\}
\]

We furthermore conjoin the expressions $\exists U. O_y$ and $\exists U. O_z$ to the left-hand side of the axiom, where $U$ is the universal role.

\[
(\exists U. O_y) \sqcap (\exists U. O_z) \sqcap \exists \text{hasReviewAssignment}.\{(a_i) \cap \{a_i\}\}
\]

\[
\sqcap \exists \text{hasSubmittedPaper}.(\exists \text{hasAuthor}.O_y \sqcap \exists \text{atVenue}.O_z)
\]

\[
\sqsubseteq \exists \text{hasConflictingAssignedPaper}.\{a_i\}
\]

\(^{10}\) This obviously needs a proof, see [51].

\(^{11}\) In this specific case, we could also simplify $((\{a_i\}) \cap (\{a_i\}))$ to $\{a_i\}$, but this is coincidental in our example.
Note that this results in $N_I$ new axioms. Finally, add to the knowledge base the following axioms, which are constructed from the safe environments and from the elements $a_i$ of $N_I$ already used:

\[
\exists U. (\{a_i\} \cap \exists \text{hasAuthor}. \{a_j\}) \subseteq \exists U. (\{a_j\} \cap O_y)
\]

\[
\exists U. (\{a_i\} \cap \exists \text{atVenue}. \{a_j\}) \subseteq \exists U. (\{a_j\} \cap O_z)
\]

Note that this results in $2 \cdot |N_I|^2$ new axioms, for a total of $|N_I| + 2 \cdot |N_I|^2$ new axioms obtained from the naive grounding—and the effect is more drastic for axioms with more nominal schemas. Note, in particular, that the number of new axioms is of the order of magnitude of $|N_I|^{\max\{2,n\}}$, where $n$ is the global bound from the definition of $\mathcal{SROELV}_n(\cap, \times)$—in particular the number is polynomially bounded for fixed $n$.

The key idea behind the transformation just described is, that the axioms (5) and (6) constrain the possible values for $O_y$ and $O_z$, and that this suffices for the reasoning process, since the concrete values obtained as elements of these concepts are not required for further processing.

4.4 Embedding Datalog under Nominal Schemas

An important feature of nominal schemas is that they can express arbitrary Datalog rules with unary and binary predicates which are interpreted as DL-safe, i.e., the predicates (and their variables) only apply to named individuals. Here, the DL-safe (Datalog) rules use a first-order logic semantics adapted using DL-safe variables—which as such is akin to a Herbrand semantics reading—which is compatible with the semantics of $\mathcal{SROIQV}(B_s, \times)$. Moreover, there is an easy syntactic transformation from DL-safe rules into $\mathcal{SROIQV}(B_s, \times)$ axioms which are semantically equivalent to the original DL-safe rules. The transformation can be done as follows:

- Each unary atom $A(x)$ is translated into $\exists U. (\{x\} \cap A)$.
- Each binary atom $R(x,y)$ is translated into $\exists U. (\{x\} \cap \exists R. \{y\})$.
- Let $B \rightarrow H$ be a DL-safe rule, $dl(H)$ be the translation of the head atom $H$, and $dl(B_i)$ be the translation of the atom $B_i$ for each atom $B_i$ in the body $B$. Then $B \rightarrow H$ is translated into $\bigwedge \{dl(B_i) \mid B_i \text{ in } B\} \subseteq dl(H)$
- Finally, the translation of a set of DL-safe rules RB is the set of axioms, each of which is the translation of an original rule from RB.

This translation clearly yields a set of axioms the size of which is linear in the size of the original rule base. Each such axiom, however, when naively grounded, results in $|N_I|^n$ new axioms without nominal schemas, where $n$ is the number of variables occurring in the originating rule. This number is exponential in $n$, however with a global bound on $n$ (as we have for $\mathcal{SROELV}_n(\cap, \times)$), it is still polynomial in the size of the knowledge base.

By way of an example, consider the rule

\[
R(x,y) \land A(y) \land S(z,y) \land T(x,z) \rightarrow P(z,x),
\]
which after the transformation defined above becomes the axiom
\[
\exists U. (\{x\} \cap \exists R. \{y\}) \\
\cap \exists U. (\{y\} \cap A) \\
\cap \exists U. (\{z\} \cap \exists S. \{y\}) \\
\cap \exists U. (\{x\} \cap \exists T. \{z\}) \\
\subseteq \exists U. (\{z\} \cap \exists P. \{x\}).
\]

4.5 Relation to OWL Profiles

Recall that OWL 2 standards have three tractable profiles for which reasoning is possible in (sub)polynomial time: OWL 2 EL, OWL 2 RL and OWL 2 QL [64]. All of them include support for datatypes and concrete data values that we omit from discussion. No technical problem will occur due to this omission as datatype literals can be treated in a similar way as individuals.

First, OWL 2 EL is contained in $\mathcal{SROEL}(\cap, \times)$ [49]. Since $\mathcal{SROEL}(\cap, \times)$ is a sublanguage of $\mathcal{SROELV}_n(\cap, \times)$ for each $n$, our approach here then subsumes the OWL 2 EL profile without datatypes.

Next, OWL 2 RL is an extension of DLP [28] and essentially based on a Horn Description Logic (see section 5.3 for discussion about DLP and Horn DL). It does neither permit disjunctive form nor existential quantification. It supports a very limited form of existential quantification, namely in such a way that it can be rewritten into a formula without existential quantification, but it includes inverse roles and unrestricted range restrictions which are disallowed in OWL 2 EL. In general, axioms of OWL 2 RL can be reduced to normal forms given below.

\[
\begin{align*}
A \subseteq C & \quad A \cap B \subseteq C & \quad R \subseteq T \\
A \subseteq \forall R.C & \quad A \subseteq \leq 1R.C & \quad R \cdot S \subseteq T \\
A \subseteq \{a\} & \quad \{a\} \subseteq C & \quad R^- \subseteq T
\end{align*}
\]

All normal forms of axioms above are clearly expressible in $\mathcal{SROELV}_n(\cap, \times)$, save for three: $A \subseteq \forall R.C$, $A \subseteq \leq 1R.C$ and $R^- \subseteq S$. But this is also not a problem because these three normal forms of axiom can be encoded using DL-safe rules which can then be translated into legal $\mathcal{SROELV}_n(\cap, \times)$ axioms in the sequel.

The normal form $A \subseteq \forall R.C$ can be encoded as the rule $A(x) \land R(x, y) \rightarrow C(y)$ which, in $\mathcal{SROELV}_n(\cap, \times)$, becomes
\[
\exists U. (\{x\} \cap A) \cap \exists U. (\{y\} \cap \exists R. \{y\}) \subseteq \exists U. (\{y\} \cap C)
\] (7)

Meanwhile, $R^- \subseteq S$ can be encoded as the rule $R(x, y) \rightarrow S(y, x)$ which can be translated into $\mathcal{SROELV}_n(\cap, \times)$ as
\[
\exists U. (\{y\} \cap \exists S. \{x\}) \subseteq \exists U. (\{y\} \cap \exists R. \{y\})
\] (8)
For $A \subseteq \leq 1R.C$, we need an auxiliary “DL-safe equality” role $R_\approx$ which is encoded using the axiom

$$\{x\} \cap \exists R_\approx \{y\} \subseteq \exists U. (\{x\} \cap \{y\})$$

We can thus encode $A \subseteq \leq 1R.C$ by the rule $A(x) \land R(x,y_1) \land C(y_1) \land R(x,y_2) \land C(y_2) \rightarrow R_\approx(y_1,y_2)$ which can be translated into $\text{SROELV}_3(\cap, \times)$ as

$$\exists U. (\{x\} \cap A) \cap \exists U. (\{x\} \cap \exists R_\approx \{y_1\}) \cap \exists U. (\{y_1\} \cap C)$$
$$\cap \exists U. (\{x\} \cap \exists R_\approx \{y_2\}) \cap \exists U. (\{y_2\} \cap C)$$
$$\subseteq \exists U. (\{y_1\} \cap \exists R_\approx \{y_2\})$$

Note that Equations (7), (8) and (9) are all legal axioms in $\text{SROELV}_3(\cap, \times)$. Thus, OWL 2 RL is subsumed by $\text{SROELV}_n(\cap, \times)$. Note however, that the translation of OWL 2 RL into $\text{SROELV}_3(\cap, \times)$ is done under DL-safe restriction. This implies that some TBox entailments are lost because the translated axioms are not semantically equivalent to the original ontology. On the other hand, if we were to allow unrestricted combination of OWL 2 EL and OWL 2 RL, we would lose tractability as reasoning becomes 2ExpTime-complete. ABox entailments, the main inference task for OWL 2 RL, are still preserved, however.

Finally, OWL 2 QL is based on DL-Lite$_R$ [8] in which inverse roles and limited forms of existential quantification are allowed, but complex RIAs are not allowed. Similar to OWL 2 RL, OWL 2 QL can be approximated using DL-safe rules, and hence by $\text{SROELV}_n(\cap, \times)$. In particular, inverse roles $R^-$ can be approximated by DL-safe rules $R_{\text{inv}}(x,y) \rightarrow R(y,x)$ and $R(x,y) \rightarrow R_{\text{inv}}(y,x)$; and axioms of the form $T \subseteq \exists R^- . C$ can be expressed as $R \subseteq \top \times C$. However, due to the use of DL-safe rules in the translation, some conclusions are lost as in the case of OWL 2 RL. Note, that the common usage of OWL 2 QL is for ontology-based querying large-scale datasets and this is possible since OWL 2 QL has a low data complexity which enables efficient query rewriting. This is obviously not supported in $\text{SROELV}_n(\cap, \times)$, although, on the other hand, it provides some features not available in OWL 2 QL, e.g., role transitivity.

5 Pointers to Further Literature

Below we discuss several other formalisms which integrate, in some fashion or other, description logics and rules. We note that there are a great many ways to achieve integration, and there are indeed multiple ways to view integration itself. Particularly, one may distinguish between syntactic integration—e.g., whether a common vocabulary is used to create rules and other sorts of assertions, and to what extent rules are syntactically isolated from other components or otherwise restricted—and semantic integration, that is whether a common semantics is used for rules and other components or whether multiple, distinct semantics are used (and then combined in some fashion). For instance, in SWRL, rules are syntactically distinct from DL axioms—there’s an ontology, and there’s also a rule base—but a uniform model theoretic semantics is used for each. In contrast,
in $\mathcal{AL}$-log, a knowledge base consists of a DL ontology and a separate Datalog program, but additionally, the semantics for each is distinct—an interpretation of a knowledge base consists of two interpretations, one for the DL ontology and another for the program. There are also formalisms where no syntactic distinction is made. That is, a common language is used (and expressions are interpreted according to a common semantics). DLP and the nominal schema formalism described in Section 4.2 fall into this category.

Along both the syntactic and semantic dimensions, there are degrees of integration—or at least considerable variation in how integration is achieved. In some cases, the syntactic and semantic separation between the sub-systems is extreme. For example, in dl-programs, a logic program is extended with atoms for interacting with an external description logic ontology, and an answer set semantics is provided for the program. But this method of interacting with a logic program is easily generalizable to other sorts of systems (i.e., non-DL systems). This is what is done in HEX-programs (which extend dl-programs).

The below list is not exhaustive, but it does describe several formalisms that are significant, either because they have been historically significant and influenced the field, or else because they indicate current research trends.

5.1 SWRL

One of the earliest formalisms combining OWL and rules is the Semantic Web Rule Language SWRL [36,37,38] (called ORL in [36]). Syntactically, SWRL extends the syntax of OWL DL and OWL Lite (circa 2004) with additional constructs to form Horn-style rule axioms. A SWRL knowledge base consists of a set of rules and OWL axioms. Semantically, the model theoretic semantics of OWL is extended to cover rules—the notable addition being the specification of variable bindings associated with interpretations.

Using an informal human readable syntax, each SWRL rule has the form $B \rightarrow H$ (as in Section 2), where $B$ and $H$ are possibly empty conjunctions of atoms. The atoms have one of the forms $C(x)$, $P(x, y)$, sameAs$(x, y)$, or differentFrom$(x, y)$, where $x$ and $y$ are variables or individuals, $P$ is an OWL property (role), and $C$ is a possibly complex OWL class (concept) description. Atoms involving datatypes and data values are also allowed, as are “built-in” atoms (for, e.g., arithmetic). We don’t discuss them here, however.

Complex class descriptions in rules can be replaced with a new class name $A$, and the two class descriptions can be declared equivalent in the OWL ontology. Similarly, sameAs and differentFrom (when it appears in the consequent of rules) can be eliminated [37].

Variables in SWRL are typed: those ranging over individuals are distinct from those ranging over data-values. Variables must also be safe, in the sense that every variable in the consequent of a rule must also appear in the antecedent. Even with this restriction, however, the satisfiability problem for SWRL knowledge bases is known to be undecidable [37].
5.2 DL-Safe Rules

The composition of rules and OWL DL\textsuperscript{12} axioms can be made decidable by forcing each rule to be DL-safe [66,73,74]. As noted above, the atoms appearing in rules may be restricted to simple unary and binary predicates (complex class descriptions can be eliminated from rules). DL-safety separates the predicates into two classes: 1) those that are names of atomic classes and roles and which are used in non-rule axioms; and 2) predicates that are not so used. Atoms making use of class and role names are called DL-atoms. A rule is DL-safe if every variable of the rule appears in a non-DL atom in the rule body. The combined knowledge base is DL-safe if every rule is. DL-safety ensures that each variable of the rule can be bound to only individuals explicitly named in the ontology.

A rule can be made DL-safe by adding, for each variable $x$ appearing in the rule, a special non-DL atom $O(x)$ to the body, and by simultaneously adding an assertion $O(a)$, for each individual name $a$, to the knowledge base. DL-safety can also be enforced by requiring each variable assignment to bind every variable to named elements in the universe of discourse. We followed the latter perspective in Section 4.1.

5.3 DLP

SWRL and DL-Safe rules do not restrict the syntax of the underlying formalisms, and DL-safety is used to ensure the decidability of the combination of rules and DL axioms. In contrast, description logic programs (DLP) [28,88] ensure decidability by restricting the formalisms to the fragment that can be expressed in def-Horn (equality- and function-free definite Horn logic) [28]. In [28], def-LP, the logic programming analog of def-Horn is also specified. The two differ in that the consequences of a def-LP program are restricted to ground atoms; no such restriction is applied to def-Horn. The atomic consequences of the program are precisely those found in the program’s least Herbrand model (which is guaranteed to exist).

Description Horn Logic is defined via a set of transformation rules to def-Horn. Specifically, the rules transform a set of DL axioms into a set of logically equivalent def-Horn rules (see Section 3.1). However, since many DL axioms yield non-Horn expressions upon transformation, certain restrictions must be made. For example, neither existential restrictions nor concept unions are permitted on the right-hand side of an inclusion axiom; universal restrictions are not allowed on the left-hand side. A Description Horn Logic ontology is simply a DL ontology whose transformation is in def-Horn. A DLP ontology is the same ontology interpreted according to the least Herbrand model semantics.

5.4 $\mathcal{AL}$-log

In SWRL, the DL axioms and rules are syntactically distinct. Nevertheless, a uniform model theoretic semantics is provided for the combination. Similarly, a

\textsuperscript{12} The papers [73,74] deal specifically with the description logic $\mathcal{SHOIN}(D)$, on which OWL DL was based; in [66] the logic used is $\mathcal{SHIQ}(D)$. 


single semantics is used for DLP. In other approaches, rules and DL systems are allowed to interact, but they are kept as distinct components (both syntactically and semantically).

In $\mathcal{AL}$-log \cite{11,12}, a knowledge base $\langle \mathcal{O}, \mathcal{P} \rangle$ is composed of an $\mathcal{ALC}$ ontology $\mathcal{O}$ (the *structural subsystem*, itself composed of an ABox and Tbox) and a Datalog program $\mathcal{P}$ (the *relational subsystem*). The Datalog program consists of *constrained* clauses: each clause $\gamma$ is accompanied by zero or more constraints $C_1(t_1), \ldots, C_n(t_n)$, where each $C_i$ is an $\mathcal{ALC}$ concept description and each $t_i$ is constant or variable. The constraints are intended to restrict the values of variables to instances of concepts. In a valid knowledge base, the following conditions must also be met: 1) the Datalog predicates of $\mathcal{P}$ are disjoint from the set of concept and role names in $\mathcal{O}$; 2) the constants of $\mathcal{P}$ coincide with the individual names of $\mathcal{O}$, and each constant of $\mathcal{P}$ appears in $\mathcal{O}$; and 3) for each constrained clause $\gamma \& C_1(t_1), \ldots, C_n(t_n)$, if $t_i$ is a variable, then $t_i$ appears in $\gamma$.

The semantics of $\langle \mathcal{O}, \mathcal{P} \rangle$ is given by providing interpretations for both $\mathcal{O}$ and $\mathcal{P}$. Let $I$ be an interpretation of $\mathcal{O}$ and $H$ a Herbrand interpretation of $\mathcal{P}$ (the constraints are ignored). $\langle I, H \rangle$ is a model of $\langle \mathcal{O}, \mathcal{P} \rangle$ if and only if $I$ is a model of $\mathcal{O}$, and for each ground instantiation of $\gamma \& C_1(t_1), \ldots, C_n(t_n)$, either there is a $C_i(t_i)$ that is not satisfied by $I$ or else $\gamma$ is satisfied by $H$. Entailment is defined in the usual fashion, save that if $a_1, \ldots, a_n$ is a set of ground atoms and $C_1(t_1), \ldots, C_m(t_m)$ a set of ground constraints, $\langle \mathcal{O}, \mathcal{P} \rangle \models a_1, \ldots, a_n \& C_1(t_1), \ldots, C_m(t_m)$ if and only if every model of $\langle \mathcal{O}, \mathcal{P} \rangle$ is a model of each $a_i$ and $C_i(t_i)$. These constitute the possible answers to queries, the latter themselves being a set of atoms together with a set of constraints. In \cite{11,12}, it is shown that query-answering for $\mathcal{AL}$-log is decidable. A query answering procedure—based on resolution—is also provided.

### 5.5 CARIN

CARIN \cite{56,57}, a family of combined DL-rule languages, is similar to $\mathcal{AL}$-log in the sense that it couples a description logic ontology to a function-free Horn-logic rule base. Unlike $\mathcal{AL}$-log, however, concept and role names are allowed to appear as predicates in rule bodies.

In \cite{56,57}, $\mathcal{ALC}NR$ is the underlying description logic used, and the problem dealt with is *existential entailment*. Two sorts of programs are examined—those with recursive rules, and those without. Without recursion, reasoning is decidable, and a sound and complete inference procedure exists. For programs with recursive rules, however, reasoning problems in CARIN-$\mathcal{ALC}NR$ are generally undecidable. Certain restrictions restore decidability, e.g. if the system employs *role safe* rules (where at least one variable of every role atom appears in a predicate that is neither in the consequent of a rule nor a concept or role name).

CARIN makes use of a classical semantics (with the unique name assumption). A single interpretation is given for both the DL ontology and rule base, and it constitutes a model of the combined knowledge base if it simultaneously satisfies both components.
5.6 \( \mathcal{DL}+log \)

\( \mathcal{DL}+log \) [78,79,80,81,82] integrates description logic ontologies with disjunctive logic programs. A \( \mathcal{DL}+log \) knowledge base is a tuple \( \langle \mathcal{O}, \mathcal{P} \rangle \), where \( \mathcal{O} \) is a DL ontology and \( \mathcal{P} \) is a logic program with rules of the form

\[
 p_1(X_1) \lor \ldots \lor p_n(X_n) \leftarrow r_1(Y_1) \land \ldots \land r_m(Y_m) \land \\
 s_1(Z_1) \land \ldots \land s_k(Z_k) \land \\
 \not u_1(W_1) \land \ldots \land \not u_h(W_h)
\]

where \( X_i, Y_i, \) etc., are tuples of variables and constants. Each \( s_i(Z_i) \) is a DL-atom (as in DL Safe rules), and every \( r_i(Y_i) \) and \( u_j(W_j) \) is a non-DL atom. The rules must be safe (every rule variable must appear in a positive literal of the body). Furthermore, every variable of the head must appear in one of the \( r_i \) atoms. This latter condition is called weak safeness. A further condition of \( \mathcal{P} \) is that it contains all constants of \( \mathcal{O} \).

\( \mathcal{DL}+log \) specifies two semantics. In the first-order semantics, the DL ontology is translated into FOL, and rules are interpreted as material implications. Negation is interpreted as classical negation. The standard names assumption is made: each interpretation is over a single countably infinite universe, each constant names the same element in each interpretation, and two distinct constants name distinct elements of the universe. In the nonmonotonic semantics, rules are interpreted according to a stable model semantics. Without negation, the two semantics yield the same results for the satisfiability problem: a knowledge base is satisfiable in one if and only if it is satisfiable in the other. In general, satisfiability for \( \mathcal{DL}+log \) KBs is decidable, provided the problem of query containment for Boolean conjunctive queries and Boolean unions of conjunctive queries is decidable in the DL used.

5.7 Horn-\( \mathcal{SHIQ} \)

Horn-\( \mathcal{SHIQ} \) [39,52,66] is a fragment of \( \mathcal{SHIQ} \) in which the ability to express disjunction has been eliminated. The definition is somewhat complicated, but Horn-\( \mathcal{SHIQ} \) knowledge bases can in general be translated into first-order Horn clauses, and every general concept inclusion axiom can be normalized into one of the below forms, where each \( A_i \) is a concept name, \( R \) and \( S \) are roles (with \( S \) simple), and \( m \geq 1 \) [15].

\[
\begin{align*}
A_i \sqcap A_j & \sqsubseteq A_k \\
\exists R.A_i & \sqsubseteq A_j \\
A_i \sqsubseteq \forall R.A_j & \quad A_i \sqsubseteq \geq m S.A_j \\
A_i \sqsubseteq \exists R.A_j & \quad A_i \sqsubseteq \leq 1 S.A_j
\end{align*}
\]

The loss of disjunction brings with it lower data-complexity. For instance, while checking satisfiability of \( \mathcal{SHIQ} \) knowledge bases (where the ABox assertions \( C(a) \) and \( \neg C(a) \) are allowed only if \( C \) is atomic) is NP-complete relative
to the size of the ABox, the problem is P-complete for similarly restricted Horn-SHIQ knowledge bases [39].

In [15], an algorithm for conjunctive query answering in Horn-SHIQ is provided. It is shown there that the entailment problem for conjunctive queries is ExpTime-complete (combined complexity). P-completeness holds for data complexity. In [40], an ExpTime algorithm for classifying Horn-SHIQ ontologies, similar in spirit to the completion based algorithm for $\mathcal{EL}^{++}$, is given.

5.8 Hybrid MKNF

Hybrid MKNF knowledge bases [65,70,71,72] combine description logics with disjunctive logic programs interpreted according to Lifschitz’s logic of minimal knowledge and negation as failure (MKNF) [58]. Formally, a Hybrid MKNF knowledge base $\mathcal{K} = \langle \mathcal{O}, \mathcal{P} \rangle$ consists of a DL ontology $\mathcal{O}$ together with a disjunctive logic program $\mathcal{P}$, where $\mathcal{P}$ is composed of DL-safe rules of the form

$$\mathbf{KH}_1 \lor \ldots \lor \mathbf{KH}_n \leftarrow \mathbf{KB}_1, \ldots, \mathbf{KB}_m, \textbf{not } C_1, \ldots, \textbf{not } C_l.$$ 

Each $H_i$, $B_j$, and $C_k$ is a function free atomic formula or else a binary formula using predicate $\approx$. The symbols $\mathbf{K}$ and $\textbf{not}$ are modal operators. Roughly, $\mathbf{K}A$ is read as “$A$ is known to hold,” and $\textbf{not } A$ as “$A$ can be false” [65].

The semantics of a Hybrid MKNF knowledge base $\mathcal{K}$ is given by translating it to a formula $\pi(\mathcal{P}) \land \mathbf{K}\pi(\mathcal{O})$ of MKNF. $\pi(\mathcal{P})$ is just the conjunction of rules of $\mathcal{P}$, each rule read as a material implication. $\pi(\mathcal{O})$ is the formula obtained by translating $\mathcal{O}$ into function-free first order logic with equality. The underlying DL must be one where such a translation is possible. The result is interpreted according to MKNF, though interpretations are restricted to Herbrand interpretations, and the standard names assumption is made.

It is noted in [70] that Hybrid MKNF generalizes several of the formalisms already discussed here, including CARIN, $\mathcal{AL}$-log, SWRL, and DL-Safe rules. Its semantics also extends both classical DL semantics and the MKNF semantics of the rules. That is, if $\mathcal{P}$ is empty, then $\mathcal{K}$’s consequences are the same as $\mathcal{O}$’s classical consequences. Similarly, if $\mathcal{O}$ is empty, then the consequences reduce to those of $\mathcal{P}$ specified by MKNF (which, as noted in [58], correspond to those determined by the stable model semantics [23,24]).

In [70,71], an algorithm for entailment checking is given, and data complexity analyses are given for knowledge bases using programs of various kinds. Without the DL-safety requirement, the satisfiability problem for Hybrid MKNF becomes undecidable.

In a separate series of papers [1,2,25,43,45,46,47], a well-founded semantics (WFS) for Hybrid MKNF knowledge bases is discussed (the rules must be normal, meaning $\neg$ does not appear). The advantage here over the semantics defined above is that it is sound relative to the original semantics but of a strictly lower complexity. Interpretations are again restricted to Herbrand interpretations, but a third truth value $u$ is added (with the ordering $f < u < t$), applicable to formulas involving modal atoms only. As above, the semantics extends both the
classical DL semantics and the traditional WFS of the rules. An alternating fix-
point procedure is defined in [44] for non-disjunctive Hybrid MKNF knowledge
bases, yielding what they call the well-founded partition.

The semantics is modified in [43,47] to ensure coherence; i.e., if \( \neg P \) holds,
then so does \( \text{not } P \). This arguably yields more intuitively correct results and
allows one to pinpoint inconsistencies. A fixpoint procedure is again defined,
and the data complexity of computing the well-founded partition is given as \( P^C \),
where \( C \) is the data-complexity of solving the ground atom entailment problem
for the underlying description logic.

A top-down method for querying Hybrid MKNF under the WFS, avoiding
the computation of the full well-founded partition, is described in [1,2]. The
method—\( SLG(O) \) resolution—alters SLG resolution [9] so that queries to an
ontology reasoner can be made. That is, the ontology reasoner is used as an
oracle. If certain restrictions are met by the oracle, then the \( SLG(O) \) method
remains tractable. A prototype reasoner (CDF-Rules), based on \( SLG(O) \) and
constructed in part using XSB Prolog, is described in [25].

5.9 dl-programs

Hybrid MKNF, like MKNF, is nonmonotonic. Another such formalism is dl-
programs [14,16,17,18,21], which again combines description logic ontologies with
extended logic programs (i.e., programs using both \( \neg \) and \( \text{not} \), the latter be-
ing default negation). The essential idea of a dl-program is that logic program
rules can contain queries to a description logic ontology. Information flow is
bidirectional—data is provided as input to the query, and answers to the queries
affect what may be inferred using the rules (which are interpreted according
to the answer-set semantics [24]). The two components are thus distinct in the
framework and yet interact in a complex way. The DLs discussed in [14] are
\( SHIF(D) \) and \( SHOIN(D) \), but the framework could be used with other DLs.

A dl-query is either a concept inclusion axiom or its negation, or else a
positive or negative concept or role assertion—e.g., \( C(t) \), \( \neg R(t_1,t_2) \), where \( C \) is
a concept description, \( R \) a role, and \( t_i \) a term. A dl-atom, which can appear in
the body of a rule but not the head, is a structure of the form \( DL[S_1 \circ p_1, \ldots, S_m \circ p_m; Q](t) \), where each \( S_i \) is a role or concept, each \( op_i \) is in the set
\( \{\cup, \cap\} \),\(^{13}\) and each \( p_i \) is a predicate from the program. Each expression \( S_i \circ p_i \)
p_i is interpreted relative to a Herbrand interpretation \( I \). \( S_i \cup p_i \) indicates that
when answering the query, atoms in the extension of predicate \( p_i \)—as specified
by \( I \)—should be included in the ontology as instances of \( S_i \). \( S_i \cap p_i \) indicates
that such atoms should be included as instances of \( \neg S_i \).

The usual notion of satisfaction by a Herbrand interpretation is extended
to apply to dl-atoms, and given this, Herbrand models for positive dl-programs
(those lacking \( \text{not} \)) are defined. Positive programs, provided they have any mo-
dels at all, have unique minimal Herbrand models which can be computed via a
fixpoint procedure. Canonical models for stratified programs are also defined.

\(^{13}\) A further operator, \( \cap \), is also discussed, but it introduces another source of non-
monotonicity even in programs without default negation. In [18], it is not discussed.
The minimal models of positive programs are used to define the answer sets of arbitrary dl-programs. Given a combined knowledge base $\mathcal{K} = \langle O, \mathcal{P} \rangle$, the **strong reduct** of program $\mathcal{P}$ relative to $\mathcal{I}$ and ontology $O$, written $s\mathcal{P}_I$, is the set of ground rules obtained by 1) deleting from the grounding of $\mathcal{P}$ all rules with an atom not $A$ in the body such that $A$ is satisfied by $\mathcal{I}$; and 2) deleting all remaining such atoms. The reduct is a positive dl-program. If its minimal model exists, then it is a **strong answer set** of $\mathcal{K}$. Without dl-atoms, every strong answer set of $\mathcal{K}$ is just an answer set of $\mathcal{P}$. Weak answer-sets, in which the reduct eliminates all dl-atoms and default negation atoms from programs, are also defined. Each weak answer set is a model of the dl-program.

If $\text{SHIF}(D)$ ontologies are used, the problem of deciding whether an unrestricted dl-program has an answer set (strong or weak) is $\text{NExpTime}$-complete. It is $\text{ExpTime}$-complete for positive and stratified programs. For $\text{SHOIN}(D)$, the problem of deciding whether a positive dl-program has a strong or weak answer set is $\text{NExpTime}$-complete. For stratified programs, it’s $\text{NP^{NExpTime}}$-complete for weak answer sets and $\text{P^{NExpTime}}$-complete for strong answer sets. For unrestricted programs, it’s $\text{NP^{NExpTime}}$-complete for both.

In [16], well-founded semantics for dl-programs are defined. The definition proceeds by first defining unfounded sets and then the operators $T_{KB}$, $U_{KB}$, and $W_{KB}$, similar to the original account of the WFS for normal logic programs [22]. An alternating fixpoint procedure for computing the well-founded model is also given, and it is shown that the semantics for dl-programs extends the WFS for normal logic programs, and also that it approximates the strong answer set semantics: every well-founded atom in the well-founded model is true in every strong answer set, and every unfounded atom is false in every strong answer set. For dl-programs based on $\text{SHIF}(D)$, determining whether a literal $l$ is in the well-founded model is $\text{ExpTime}$-complete. For $\text{SHOIN}(D)$, the corresponding problem is $\text{P^{ExpTime}}$-complete.

Observe that dl-queries essentially provide an interface between a logic program and a distinct DL ontology. This basic framework permits the use of external data sources other than DLs. This is the basic idea behind **HEX-programs** (higher order logic programs with external atoms) [19,20,86]. Disjunctions are allowed in the heads of rules, and instead of dl-atoms, programs make use of external atoms of the form $\&[Y_0(Y_1,\ldots,Y_n)](X_1,\ldots,X_m)$, where $g$ is an external predicate (not used save in such atoms) and $[Y_0(Y_1,\ldots,Y_n)]$ and $(X_1,\ldots,X_m)$ are input and output lists of terms, respectively. A solver for HEX-programs, dlhex, has been implemented (by extending the answer-set solver dlv15).

### 5.10 Disjunctive dl-programs

Another formalism [59] also goes by the name “dl-programs”, but it is unrelated to the formalism described above. In [59], a knowledge base $\langle O, \mathcal{P} \rangle$ is again

---

14 The programs are normal in the sense that negative literals $\neg a$ are not allowed. Furthermore, the semantics is only defined for dl-programs not involving $\cap$.

15 http://www.dbai.tuwien.ac.at/proj/dlv/
formed by combining a (disjunctive) logic program $\mathcal{P}$ with a DL ontology $\mathcal{O}$, but in this case the logic program is a more typical disjunctive logic program (i.e., there are no dl-atoms). Only one form of negation, default negation, is allowed. Constants of the program are a subset of the individuals in the DL ontology, but no other special restrictions are made on the vocabulary used.

A uniform semantics is used. The basic idea is to interpret $\mathcal{P}$ using Herbrand interpretations that also satisfy $\mathcal{O}$. That is, a Herbrand interpretation $\mathcal{I}$ of a program $\mathcal{P}$ is any subset of the Herbrand base $HB$ of the program. $\mathcal{I}$ is a model of $\mathcal{O}$ if and only if $\mathcal{O} \cup \mathcal{I} \cup \{\neg a | HB - \mathcal{I}\}$ is satisfiable. $\mathcal{I}$ is a model of $\langle \mathcal{O}, \mathcal{P} \rangle$ if $\mathcal{I}$ models both $\mathcal{P}$ and $\mathcal{O}$. $\mathcal{I}$ is an answer set of $\langle \mathcal{O}, \mathcal{P} \rangle$ if it is a minimal model of $\langle \mathcal{O}, \mathcal{P}^I \rangle$, where $\mathcal{P}^I$ is the reduct of $\mathcal{P}$ with respect to $\mathcal{I}$.

The semantics described above extends the answer set semantics for disjunctive logic programs: If $\mathcal{O}$ is empty, then the answer sets for $\langle \mathcal{O}, \mathcal{P} \rangle$ are the answer sets for $\mathcal{P}$. If instead $\mathcal{P}$ is empty, a ground atom $a$ is true in every answer set of $\langle \mathcal{O}, \mathcal{P} \rangle$ if and only if it is true in all first-order models of $\mathcal{O}$. It is shown in [59] that, if $\mathcal{O}$ is in SHLFD($D$) or SHOLN($D$), then deciding whether the combined knowledge base has an answer-set is NEXP$^\text{NP}$-complete. Determining whether a ground atom $a$ is true in all (some) answer-sets of the knowledge base is $\text{co-NEXP}^\text{NP}$-complete (NEXP$^\text{NP}$-complete). Reasoning algorithms for deciding the existence of answer-sets are also identified, as is a class of stratified knowledge bases (based on DL-Lite). For such knowledge bases, the problems of deciding whether an answer set exists (which must be unique, if it exists), and whether a given ground atom is true in it, have polynomial data-complexity.

5.11 Quantified Equilibrium Logic for Hybrid Knowledge bases

In [10], it is shown how a variation of the Quantified Equilibrium Logic (QEL) [77] can be used as a semantics for hybrid knowledge bases, one which encompasses other semantics proposed in the literature. Here, a hybrid knowledge base is defined to be a combination $\langle \mathcal{O}, \mathcal{P} \rangle$ of first order theory $\mathcal{O}$ and a disjunctive logic program $\mathcal{P}$. $\mathcal{P}$ may contain first order literals $a$ and $\neg a$. Both components are function-free and are defined using the same constants. $\mathcal{P}$’s predicates are a superset of $\mathcal{O}$’s. The stable closure of a hybrid knowledge base is defined (essentially by taking the union of $\mathcal{O}$ and $\mathcal{P}$ and adding $(\forall X)(p(X) \lor \neg p(X))$ for each predicate of $\mathcal{O}$), and equilibrium models are then defined for the stable closure. It is shown that by varying restrictions on the domain of discourse, these models correspond to models of the hybrid knowledge base according to frameworks proposed by Rosati, including $\mathcal{DL}+\text{log}$ (discussed above), and according to guarded-hybrid (g-hybrid) knowledge bases [29].

5.12 Description graphs

Description graphs [63,67,68,69] extend DLs with first-order rules and graphs allowing the representation of structured objects (such as the bones of a hand) not otherwise expressible in a DL. The graphs can be arranged into a hierarchy (which may be used to describe an object at differing levels of granularity).
In the framework, an \( n \)-ary description graph \( G \) is a directed graph of \( n \) vertices, with each vertex labeled with a set of atomic concepts or their negations, and each edge labeled with a set of atomic roles or their negations. Some subset of the atomic concepts is selected as constituting the main concepts of the graph (roughly, they indicate what the graph is about). A graph specialization axiom \( G \sqsubseteq G' \) indicates that each vertex of \( G \) is one of \( G' \). A graph alignment axiom \( G_1[v_1, \ldots, v_n] \leftrightarrow G_2[u_1, \ldots, u_n] \) is a 1-1 mapping of some subset of vertices of two graphs. A graph box (GBox) \( G \) is a finite collection of description graphs, specialization axioms, and alignment axioms. A graph assertion is an expression of the form \( G(a_1, \ldots, a_n) \), where \( G \) is an n-ary description graph and each \( a_i \) is an individual.

The bodies of rules consist of conjunctions of atomic concept atoms \( C(t) \), atomic role atoms \( R(t_1, t_2) \), but also graph atoms \( G(t_1, \ldots, t_k) \), where each \( t_i \) is an individual or a variable and \( G \) is a description graph. Rule heads are disjunctions of such atoms (the head may also contain equality atoms \( t_1 \approx t_2 \)). Each rule must be connected: for any variables \( x \) and \( y \) in the rule, there is a sequence \( x_1, \ldots, x_n \) of variables such that \( x_1 = x \) and \( x_n = y \) and for each \( i < n \), \( x_i \) and \( x_{i+1} \) appear in the same body atom.

A graph extended knowledge base is a tuple \( K = (T, P, G, A) \), where \( T \) is a TBox, \( P \) is a finite set of connected graph rules, \( G \) is a GBox, and \( A \) is an ABox possibly containing graph assertions. In an interpretation \( I \), each \( n \)-ary graph \( G \) is read as an \( n \)-ary relation over \( \Delta^I \). An assertion \( G(a_1, \ldots, a_n) \) is satisfied by \( I \) if and only if \( (a_1^I, \ldots, a_n^I) \in G^I \). The semantics is such that in any model of \( K \), no two distinct instances of a description graph share vertices, and the vertices are ensured to participate in the concepts and role relations indicated in the graph. \( G \sqsubseteq G' \) holds if each instance of \( G' \) is an instance of \( G \), and \( G_1[v_1, \ldots, v_n] \leftrightarrow G_2[u_1, \ldots, u_n] \) holds if, whenever instances of \( G_1 \) and \( G_2 \) share vertices \( u_i \) and \( v_i \), then they share all other vertex pairs in the axiom.

Under many circumstances, the satisfiability problem for graph extended knowledge bases is undecidable—for example, if \( T \) is empty, \( P \) is Horn, and no specialization or alignment axioms are used. Decidability can be regained in this example by requiring the hierarchy of graph descriptions to be “acyclic” (see [63]). In other cases, however, additional restrictions are required. In [63], it is shown that the satisfiability problem for an acyclic \( K \) is \( \text{NExpTime} \)-complete, provided \( K \) is weakly separated and \( T \) is in \( \text{SHOQ}^+ \). Alternatively, it is \( \text{NExpTime} \)-complete if \( K \) is strongly separated and \( T \) is in \( \text{SHIQ}^+ \). Here, weak separation means that the roles of \( T \) and \( P \) are disjoint. Strong separation additionally requires the roles of \( T \) to be disjoint with those of \( G \).

6 Conclusions

We have reported on the considerable body of work on OWL and Rules, describing integration proposals that sometimes differ substantially in terms of their underlying approach and rationale. Some approaches have been more popular than others. In some cases, it appears to be a matter of subjective judgement
regarding which provide the best underpinnings for a “unified logic” in the sense of the W3C Semantic Web Stack. And it’s likely additional alternatives will be proposed in the future. that we will see a few more alternative proposals in the near future.

Further theoretical investigations will certainly shed more light on the issue. Concerning the proposed formalism in Section 4, for example, it would be helpful to investigate possibilities for incorporating nonmonotonic negation or other closed world features [3,7,13,26,27,47,48,65,71] which commonly occur in logic-programming-based rule approaches. For example, we have recently proposed an intuitively appealing approach for extending description logics with local closed world features which retains decidability if added to the description logics with nominal schemas discussed herein [48]. Even more importantly, however, efficient algorithms and implementations need to be developed.

In the end, usability aspects will also play a decisive role, and it is here where the development of Semantic Web applications involving deep reasoning are often found to be lacking [31,32]. The Semantic Web requires usable tools, interfaces, design patterns, and best-practice guidelines which would allow developers to use ontologies and underlying reasoning paradigms without having to become expert logicians. We’re still a long way away from that goal.

Acknowledgements This work was supported by the National Science Foundation under award 1017225 III: Small: TROn—Tractable Reasoning with Ontologies. Adila Krisnadhi acknowledges support by a Fulbright Indonesia Presidential Scholarship PhD Grant 2010.

References


http://www.w3.org/2007/03/layerCake.png
See, e.g., [35,76]


general logic programs. In: PODS ’88: Proceedings of the seventh ACM SIGACT-
ACM Press (1988)

23. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In:
Kowalski, R.A., Bowen, K. (eds.) Proceedings of the Fifth International Conference
(1988)

24. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive

MKNF Knowledge Bases. In: Carro, M., Peña, R. (eds.) Practical Aspects of
Springer Berlin / Heidelberg (2010)

126 (2008)

27. Grimm, S., Hitzler, P.: A Preferential Tableaux Calculus for Circumscriptive
ALCO. In: Polleres, A., Swift, T. (eds.) Proceedings of Third International
Conference on Web Reasoning and Rule Systems, Chantilly, VA, USA, October 25-26,
(2009)

28. Grosof, B., Horrocks, I., Volz, R., Decker, S.: Description Logic Programs: Com-

knowledge bases. In: Proc. of ICLP’06 Workshop on Applications of Logic Pro-


31. Hitzler, P.; Towards reasoning pragmatics. In: Janowicz, K., Raubal, M., Levashkin,
S. (eds.) GeoSpatial Semantics, Third International Conference, GeoS 2009, Mex-
Computer Science, Springer (2009)


2 Web Ontology Language: Primer. W3C Recommendation 27 October 2009
(2009), available from http://www.w3.org/TR/owl2-primer/

Chapman & Hall/CRC (2009)

35. Hitzler, P., Seda, A.K.: Mathematical Aspects of Logic Programming Seman-
tics. CRC Press (2010)

36. Horrocks, I., Patel-Schneider, P.F.: A proposal for an OWL rules language. In:

37. Horrocks, I., Patel-Schneider, P.F., Bechhofer, S., Tsarkov, D.: OWL rules: A pro-
posal and prototype implementation. J. of Web Semant. 3, 23–40 (July 2005)


85. Rudolph, S., Krötzsch, M., Hitzler, P.: All elephants are bigger than all mice.
In: Baader, F., Lutz, C., Motik, B. (eds.) Proceedings of the 21st International
Vienna University of Technology, Austria (2006)
87. Tsarkov, D., Sattler, U., Stevens, R.: A solution for the Man-Man problem in
the Family History Knowledge Base. In: Hoekstra, R., Patel-Schneider, P.F. (eds.)
Proceedings of the 5th International Workshop on OWL: Experiences and Direc-
tions (OWLED 2009), Chantilly, VA, United States, October 23-24, 2009. CEUR
Fridericana zu Karlsruhe (TH), Germany (2004)