Combining OWL with F-Logic Rules and Defaults

Heiko Kattenstroth, Wolfgang May, and Franz Schenk

Institut für Informatik, Universität Göttingen,
{hkattens|may|schenk}@informatik.uni-goettingen.de

Abstract. We describe the combination of OWL and F-Logic for the architecture of Semantic Web application nodes. The approach has been implemented by combining an existing Jena-based architecture with an external Florid instance. The approach provides a tight language coupling, i.e., the same notions can be defined both by OWL definitions and by F-Logic rules. F-Logic rules are used for e.g., role-value-maps, closed-world-reasoning, (stratified) negation, aggregation, and definition of answer views; additionally the default inheritance of F-Logic can be exploited.

1 Introduction

Description Logics [BCM+03] provide the underlying base for the dominating data model and languages for the Semantic Web, RDF, RDFS, and OWL [OWL04]. It is based on the notion of classes, auch as Person, Country, City, and properties, auch as hasName, hasChild, livesIn, hasCapital. This corresponds to unary and binary relations in First-Order-Logic and relational databases, such as Person(john), hasName(john,"John"), and livesIn(john,berlin). Description Logics additionally allow for further specifications of classes (and properties) that have no direct equivalent in relational databases, e.g. that a parent is a person who has at least one child: $\mathsf{Parent} \equiv \mathsf{Person} \sqcap \exists \mathsf{hasChild}.\top$ or the assertion that children are persons, $\mathsf{Parent} \sqsubseteq \forall \mathsf{hasChild}.\mathsf{Person}$. Such things can be expressed by rules in FOL, but are not inherent concepts of FOL semantics.

Thus, the "built-ins" are more advanced than in FOL, but on the other hand, DL formulas are much restricted to concept expressions. The application of conjunction, and even more disjunction and negation, is only allowed in terms of these built-ins. The Description Logic $\mathcal{SHOIQ}(\mathcal{D})$ [HS05] that forms the base for OWL-DL is decidable, but there are "simple" concepts that are still out of reach in this fragment, e.g., composite properties such as "uncle" as "brother of parent" cannot be expressed.

The combination of Description Logics with rules is thus a prominent research topic, e.g., investigated early in $\mathcal{AL}$-*log* [DLNS91,DLNS98], CARIN [LR96], or more recently in numerous approaches, e.g., DLP [GHVD03], SWRL (earlier: ORL) [HPS04,HPSB+04], DLV/DLVhex [ELST04,EIST06], DL+log [Ros06], OWL-Flight [dBLPF05], *DL-safe* rules [MSS05], [Luk07] or [DM07] (see Section 5 for a more detailed analysis).

The goal of our approach is primarily pragmatic: to provide an architecture for an application service node in the Semantic Web (e.g., hosting the information system of an airline service or a university) and providing appropriate interfaces to the outside. For that, we combine Description Logic with (full) F-Logic, which also brings *default inheritance* into play.

DL+Florid provides a *tight* coupling in the sense that the symbols of the DL part and the rule part are not required to be disjoint. Thus, the rules can be used (and are intended) to derive concept memberships and role instances.

The semantics of DL+Florid is defined in a bottom-up way (that is also realized by the implementation that combines the Jena [Jen] Framework with a plugged-in Pellet reasoner [Pel] and Florid 4.0 [FHK+97,FLO06]). The Jena-based DL system provides the core of the architecture that employs Florid as a "slave" for rules and default inheritance. After reviewing the basic notions in Section 2, we describe the architecture and analyze the semantics of our pragmatic approach in Section 3. Inheritance based on defaults is then discussed in Section 4. Section 5 gives a comparison with related work, and Section 6 concludes the paper.

2 Basics

2.1 Overview of Description Logics and RDF/RDFS/OWL

Description Logics (DLs) [BCM+03] are a family of logics for concept reasoning. Their main constructs are classes and properties, expressed by (i) class membership atoms, e.g., $C(a)$ (object a is an instance of class C), property atoms $p(a,b)$ (b is some value of property p of object a), subclass axioms $C \sqsubseteq D$, and class equivalence $C \equiv D$. Different DLs allow or disallow certain constructs for describing class definitions. The current focus is on decidable DLs where complete decision procedures exist. DLs are the underlying framework for the Semantic Web languages RDF, RDFS, and OWL [OWL04] with its variants OWL-Lite, OWL-DL and OWL-Full. For OWL-DL, currently extensions to OWL-1.1 are under discussion. OWL-DL is based on the decidable Description Logic $\mathcal{SHOIQ}(\mathcal{D})$ [HS05]; the extensions belong to $\mathcal{SROIQ}$ which allows additional concepts for specifying properties [HKS06]. As DLs are a restricted fragment of First-Order Logic, FOL model theory and semantics applies to them, which means that in contrast to Logic Programming, *open-world* semantics applies. Reasoners like Pellet [Pel] support OWL-1.1.

2.2 Overview of F-Logic

As stated above, the DL+Florid approach extends DL with deductive rules and default inheritance. Both are features that are natively supported by F-Logic and its implementation in Florid [FHK+97,FLO06]. F-Logic rules are logic programming rules over F-Logic atoms. F-Logic atoms are defined as follows (cf. [KLW95]); we use only properties without parameters (i.e., only the form $o[m{\rightarrow}v]$, not $o[m@(a_1,\ldots,a_n){\rightarrow}v]$).

Definition 1 (Syntax of F-Logic). *The alphabet of an F-Logic language consists of a set $\mathcal{F}$ of* function symbols*, playing the role of object constructors. For convention, function symbols start with lowercase letters whereas variables start with uppercase ones.* Id-terms *are composed from object constructors and variables and are interpreted as elements of the universe.*

In the sequel, let o, c, $d, d_1, \ldots, d_n$, p, $v, v_1, \ldots, v_n$ stand for id-terms or literals. Note that URLs as a subclass of strings can denote objects; e.g.

$$\text{“foo:bla\#john”}:\text{“foo:meta\#Person”}\,[\text{“foo:meta\#name”}\rightarrow\text{“John”};$$
$$\text{“foo:meta\#livesIn”}\twoheadrightarrow(\text{“geo://de/Berlin”}:\text{“geo:meta\#City”})].$$

is a valid F-Logic fragment; see also later examples.

1. *An* is-a atom *is an expression of the form $o : c$ (object o is a member of class c), or $c :: d$ (class c is a subclass of class d).*
2. *The following are* object atoms*:*

2a. *$c[p \Rightarrow (d_1, \ldots, d_n)]$ and $c[p \Rrightarrow (d_1, \ldots, d_n)]$: the values of the* scalar *or* multivalued*, respectively, property p of objects of class c belong (simultaneously) to all classes $d_1, \ldots, d_n$,*

2b. *$o[p \rightarrow v]$: the* scalar *property p of object o has the value v,*

2c. *$o[p \twoheadrightarrow \{v_1, \ldots, v_n\}]$: $\{v_1, \ldots, v_n\}$ are amongst the values of the* multivalued *property p of object o,*

2d. *$c[p \bullet\!\!\rightarrow v]$: for objects of class c, the default value of the* scalar *property p is v.*

2e. *$c[p \bullet\!\!\twoheadrightarrow \{v_1, \ldots, v_n\}]$: for objects of class c, the default values of the* multivalued *property p are $\{v_1, \ldots, v_n\}$.*

An F-Logic rule *is a logic rule $h \leftarrow b$ over F-Logic atoms, i.e. is-a assertions and object atoms. An F-Logic* program *is a set of rules.*

The semantics of F-Logic rules and defaults is defined via Herbrand-style structures where the universe consists of ground id-terms. An *H-structure* is a set of ground F-Logic atoms describing an object world, thus it has to satisfy several *closure axioms* related to general object-oriented properties:

Definition 2 (F-Logic Axioms). *A (possibly infinite) set $\mathcal{H}$ of ground atoms is an* H-structure *if the following conditions hold for arbitrary ground id-terms u, $u_0, \ldots, u_n$, and u_m occurring in $\mathcal{H}$:*

- *$u :: u \in \mathcal{H}$ (subclass reflexivity),*
- *if $u_1 :: u_2 \in \mathcal{H}$ and $u_2 :: u_3 \in \mathcal{H}$ then $u_1 :: u_3 \in \mathcal{H}$ (subclass transitivity), analogously, if $u_1 : u_2 \in \mathcal{H}$ and $u_2 :: u_3 \in \mathcal{H}$ then $u_1 : u_3 \in \mathcal{H}$,*
- *if $u_1 :: u_2 \in \mathcal{H}$ and $u_2 :: u_1 \in \mathcal{H}$ then $u_1 = u_2 \in \mathcal{H}$ (subclass acyclicity),*
- *if for ground id-terms u and u' ($u \neq u'$) such that $u_0[u_m \rightsquigarrow u] \in \mathcal{H}$ and $u_0[u_m \rightsquigarrow u'] \in \mathcal{H}$, then $u = u'$, where $\rightsquigarrow$ stands for $\rightarrow$ or $\bullet\!\!\rightarrow$ (uniqueness of scalar properties).*

For a set M of ground atoms, $\mathcal{Cl}(M)$ denotes the closure of M wrt. the above axioms.

Positive F-Logic programs are evaluated bottom-up by a T_P-like operator including $\mathcal{C}\ell$, providing a minimal model semantics:

Definition 3 (Deductive Fixpoint).
For an F-Logic program P and an H-structure $\mathcal{H}$,

$$\begin{aligned}
T_P(\mathcal{H}) &:= \mathcal{H} \cup \{h \mid (h \leftarrow b_1, \ldots, b_n) \textit{ is a ground instance of some rule of } P \\
&\qquad\qquad \textit{and } b_i \in \mathcal{H} \textit{ for all } i = 1, \ldots, n\}\ , \\
T_P^0(\mathcal{H}) &:= \mathcal{C}\ell(\mathcal{H})\ , \\
T_P^{i+1}(\mathcal{H}) &:= \mathcal{C}\ell(T_P(T_P^i(\mathcal{H})))\ , \\
T_P^\omega(\mathcal{H}) &:= \begin{cases} \lim_{i\to\infty} T_P^i(\mathcal{H}) & \textit{if the sequence } T_P^0(\mathcal{H}), T_P^1(\mathcal{H}), \ldots \textit{ converges,} \\ \perp \ \ \textit{otherwise.} \end{cases}
\end{aligned}$$

User-stratified programs are evaluated analogously wrt. Perfect Model semantics. The above semantics that covers deductive rules does not deal with inheritance; this will be described in Section 4.

Correspondence with DL and RDF/RDFS/OWL. In an RDF/OWL setting, objects are identified by URIs and by ids of blank nodes; thus, for theoretical considerations, the restriction to function-free F-Logic is reasonable. Note that in F-Logic, id-terms and objects also stand for properties, in the same way as URIs in RDF. Variables can also occur at arbitrary positions of an atom.

The *isa-atoms* (1) correspond to DL's $C(o)$ and $C \sqsubseteq D$ (i.e., rdf:type and rdfs:subclass). The object atoms (2a) correspond to $C \sqsubseteq \forall p.D$ (i.e., rdfs:range), (2b) and (2c) correspond to the DL property assertions $p(o, v)$ (i.e., the RDF triple (o, p, v)). The inheritance atoms (2d) and (2e) have no equivalent in DL or RDF/OWL. Together with the rules, the semantics of (2d) and (2e) provide the additional expressiveness of DL+Florid.

2.3 Why Rules?

Hybrid approaches that combine DL with deductive rules are of interest for several reasons:

Higher Expressiveness: Positive Rules. Many things that cannot be expressed in OWL can be defined easily with rules, even with often only *positive* rules. These are e.g., composite roles that do not satisfy the *tree property* (although restricted support comes with OWL 1.1), or annotated and computed properties. For instance, connection(city_1, city_2) is transitive and thus can be expressed in OWL, but as such connections are represented by *role instances*, they cannot have properties like distance (except by reification). Even with reification, it cannot be expressed in OWL that the distance of composite connections is obtained by adding the individual distances (see Example 3 later). Furthermore, aggregations like count, sum, max, avg can be expressed in most rule languages (but these are then actually not just positive rules).

Apart from the above expressiveness issues, logical rules with an operational flavor are often used for ontology integration and data integration.

Higher Expressiveness: Negation under CWA. Another issue is the use of default negation (often also called "negation as failure" which is actually its implementation in Prolog): facts that are not explicitly known are assumed not to hold. This is relevant e.g. when dealing with unmarried or childless people, countries without big cities etc. While "unmarried" is still a property that is often explicitly given, concepts like "country without big cities" are usually to be derived. Default negation also underlies the definition of aggregations, since these implicitly also assume that no additional facts have to be taken into account for the aggregation.

Query Answering. Rules allow for a declarative *and* constructive specification how a result can be obtained. For function-free normal and stratified rules, evaluation of queries (=views) is polynomial.

Example 1 (Rules for Query Answering). Consider the simple geographic ontology of the Mondial database [May07], containing (among others) the notions

```
Classes: Country, Province, City,
Properties: hasProvince, isProvinceOf, hasCity, cityIn, population.
```

For some countries, no provinces are known and the cities are directly associated with the countries; for the others, cities are associated with the provinces.

Compute all countries that have at least two cities with more than 1.000.000 inhabitants. In an OWL ontology, a composite relationship between countries and cities covering the hierarchy has to be defined (using transitivity), and concepts (as restrictions) BigCity and CountryWithTwoBigCities must be defined:

```
:isProvinceOf rdfs:subpropertyOf :belongsTo.
:cityIn rdfs:subpropertyOf :belongsTo.
:belongsTo a owl:TransitiveProperty;
    owl:inverseOf :hasProvOrCity.     ## bridge country-prov-city
:Million a owl:DataRange; owl11:onDataRange xsd:int; owl11:minInclusive 1000000.
:HasBigPopulation owl:equivalentClass [a owl:Restriction;
    owl:onProperty :population; owl:someValuesFrom :Million].
:BigCity owl:intersectionOf (:City :HasBigPopulation).
:CountryWithTwoBigCities owl:intersectionOf (mon:Country
  [a owl:Restriction; owl:onProperty :hasProvOrCity;
    owl:minCardinality 2; owl11:onClass :BigCity]).
```

The actual evaluation by a reasoner takes some minutes. In contrast, with (positive) rules, the same can be defined:

```
## (note: use c_Classname for Classes)
Cty:c_BigCity :- Cty:c_City, Cty[population→→Pop], Pop > 1000000.
C[hasBigCity→→Cty] :- C[hasCity→→Cty], Cty:c_BigCity.
C[hasBigCity→→Cty] :- C[hasProvince→→Prov], Prov[hasCity→→Cty], Cty:c_BigCity.
X:c_CountryW2BigCities :- X:c_Country, X[hasBigCity →→{C1,C2}], not C1 = C2.
```

where evaluation is significantly faster. With stratified negation, also e.g. all countries that have no big cities can be listed.

The above example shows that already a "decoupled" combination of an OWL core with rule-based views for computing answers provides certain advantages. Similarly, rules can be used to define concepts although they could be defined as owl:Restrictions, not only as answer views, but also for efficiency.

3 Combining DL+Florid

The goal of the approach is to provide an architecture for an application service node in the Semantic Web (e.g., hosting the information system of an airline service or a university) and providing appropriate interfaces to the outside. Thus, we are primarily interested in a pragmatic approach.

The architecture is shown in Figure 1. The node core is based on the Jena framework [Jen]. It contains a database (e.g. PostgreSQL) as repository for the ontology (i.e., concepts and properties), the facts, and also for the rules. It can be queried with SPARQL [SPQ06] as the current mainstream Semantic Web language. For OWL reasoning, an instance of the Pellet DL reasoner [Pel] is connected to the node. This basic functionality has been extended with a simple update language for RDF data and with support for RDF-level database triggers reacting upon *database update actions*, e.g. to support actual updates when deleting an instance $p(x, y)$ of a property that is symmetric and stored as $p(y, x)$ [MSvL06]. Additionally, a Florid instance is connected as a "slave" for application of F-Logic rules and default reasoning.

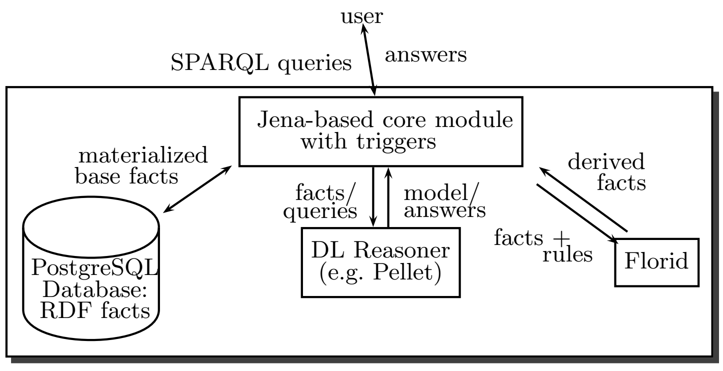

Fig. 1. Architecture of DL+Florid

3.1 Reasoning with DL+Florid

The general idea is to separate concerns into (i) OWL concept (TBox) reasoning, (ii) application of rules, and (iii) inheritance. Reasoning about facts can be either done in the OWL portion (ABox; e.g., transitivity, inverse, symmetry), or by rules, which allow for much more complex derivations. The knowledge base $KB = (L, P, D)$ is thus partitioned into

- an OWL ontology L,
- a finite set P of rules (in F-Logic or a RuleML-style XML markup), and
- a finite set D of inheritance atoms (defaults).

For things that can alternatively specified by OWL or F-Logic (mostly, class membership characterizations), the *alternating fixpoint* evaluation described below guarantees the same outcome under certain conditions.

This means that usually rule bodies contain only domain notions, and no RDF/RDFS/OWL properties (mainly, to reduce the amount of data to be transferred – using these properties is the native responsibility of the OWL reasoner). Rule *heads* are allowed to contain RDF/RDFS/OWL notions, although this seems to be an unusual case (e.g., used for concept learning).

The evaluation proceeds as follows, taking the OWL ontology in the Jena-based node as starting point:

First Step: Compute the OWL model of a given fact base. Due to its open-world nature, OWL/DL reasoning contains only limited negation. Anything derived *later* will be taken into account as "possible". Doing OWL reasoning first is thus completely safe.

Second Step: Application of deductive rules. All (relevant) facts are exported together with the rules to Florid, where bottom-up-evaluation is applied; the (relevant) resulting facts are sent back to the Jena-based core. In case of positive rules, it is again completely safe wrt. facts derived in later steps by OWL reasoning (see below for a discussion).

Iteration: the above steps are iterated until no new facts can be derived.

Inheritance Step: only when both OWL reasoning and application of rules are not able to derive further facts, default inheritance takes place.

Iteration: As long as new facts have been derived by default inheritance, the above inner iteration is restarted. Note that this corresponds to the F-Logic semantics of default inheritance – applying default inheritance only after the application of rules reached a fixpoint does not derive any new facts, and then restarting the iteration – that has been shown in [MK01] to be "reasonable" and compatible with the Default Logic [Rei80,Poo94] semantics.

3.2 Data Exchange and Handling of URLs

Whereas the evaluation of the OWL specification is based on theory reasoning, the evaluation of rules is based on *ground facts* (and non-ground rules).

Export of facts to Florid: The Jena *model* containing the (base and derived) facts is dumped, and triples are exported as atoms as follows: x rdf:type c where c is an application class (as x_t:c_t), c rdfs:subClassOf d (as c_t::d_t), p rdfs:range

c where p is a non-RDF/RDFS/OWL property (as <owl:Thing>$_t[p_t{\Rightarrow}c_t]$ (if p is known to be functional) or <owl:Thing>$_t[p_t{\Rightarrow\!\!\!\Rightarrow}c_t]$ (otherwise)), $x_t\ p_t\ y_t$ where p is a non-RDF/RDFS/OWL property (as $x_t[p_t{\rightarrow}y_t]$ (if p is known to be functional) or $x_t[p_t{\twoheadrightarrow}y_t]$ (otherwise)).

Above, the mapping $*_t$ of the identifiers is as follows: for literals ℓ, i.e., strings and numbers, ℓ_t is the string representation of ℓ, e.g., "bla", 1, or 3.1415. URIs (including ids of blank nodes) are exported as elements of the class url::string, e.g., "http://www.w3.org/2002/07/owl#Thing":url (the distinguished class url::string for URLs has been used for accessing HTML documents in earlier times).

Example 2. For example, the N3 data

```
<foo:meta#Person> a owl:Class.
<foo:meta#name> a owl:FunctionalProperty; a owl:DatatypeProperty.
<foo:meta#livesIn> a owl:FunctionalProperty; a owl:ObjectProperty.
<foo:bla#john> a <foo:meta#Person>; <foo:meta#name> "John";
               <foo:meta#livesIn> <geo://de/Berlin>.
```

is translated into F-Logic:

```
"foo:meta#Person":url.    "foo:meta#name":url.    "foo:meta#livesIn":url.
"foo:meta#john":url.      "geo://de/Berlin":url.
("foo:bla#john":"foo:meta#Person")["foo:meta#name"→"John";
          "foo:meta#livesIn"↠"geo://de/Berlin"].
```

Export of facts back to Jena: On the way back, the above translation is inverted. Here, only atoms/triples $x{:}c$, $c{::}d$, $x[p{\rightarrow}y]$, and $x[p{\twoheadrightarrow}y]$ are exported where x, c, d, p are members of the class url; y may be an url or a literal. This allows to use auxiliary predicates, classes and identifiers in the rule part that are not exported back to Jena. Note that it is also possible to introduce new classes, properties, and objects by urls in the rule part.

Back in Jena, the returned data is merged with the before data; if new facts have been derived, the alternating process is iterated until a fixpoint is reached.

3.3 Positive Rules

If the program P only consists of positive rules, everything is safe. Independent how many (inner – i.e., between Jena and Rules) iterations are executed until a fixpoint is reached, the resulting structure is a subset (cf. Section 3.5) of what would be derived when using e.g. [ELST04,EIST06].

3.4 Rules with Negation

Florid supports user-defined stratification of programs. Concerning the interference of iterating OWL reasoning and rule application, the CWA of LP negation conflicts with the possibility of later derivation of positive facts by OWL reasoning in the above fixpoint process. The evaluation of stratified programs in the given alternating combination with OWL reasoning is correct wrt. the *stratified/perfect model semantics* if the extension of predicates that occur negatively in a rule is not changed after evaluating it for the first time (i.e., also not in subsequent OWL rounds). Operationally, this condition can be verified by combining syntactical analysis of rules with runtime monitoring:

– Let Σ^- denote the set of all predicates occurring negatively in a rule body.
– Let $\mathcal{I}$ denote the interpretation after the first iteration of computing the OWL model and applying the rules once (without applying OWL reasoning again). Let $\mathcal{I}_{neg} := \mathcal{I}|_{\Sigma^-}$ ($\mathcal{I}$ restricted to Σ^-).
– For each iteration, check if for the current interpretation $\mathcal{I}'$, $\mathcal{I}'|_{\Sigma^-} = \mathcal{I}_{neg}$.

This allows e.g. to use the negation of all base facts. Since classes in Σ^- often occur in rdsf:range and rdsf:domain axioms, pure ontology analysis will in most cases derive that it is *possible* that they could be extended during reasoning.

An evaluation that is correct in the general case in presence of negation in the rules is only possible in a tightly coupled evaluation in the DL reasoner (which then requires to restrict to DL-safe rules).

Rules for Views and Query Answering. The above condition is trivially satisfied when the predicates in the rule head are disjoint from those used in the OWL part, but occur only in the rule program.

3.5 Pitfalls: Existential Assertions in OWL

In the investigations of hybrid rules, it turned out that the crucial problem are anonymous, implicit objects (see Section 5) that affect decidability (note: these are not the blank nodes, but purely existential objects as in $\mathsf{Parent} \equiv \mathsf{Person} \sqsubseteq \exists \mathsf{hasChild.Person}$, or in $\mathsf{Person} \sqsubseteq \exists \mathsf{hasFather.Person}$). The constraints on variables developed over time aimed at restricting variables such that they cannot be bound to these anonymous objects; or, taken the other way round [Ros06]: every *head* variable must occur in an LP atoms in the body (such that only objects from the explicit active domain can be bound to variables that occur in the head). Actually, this prevents from deriving anything about these anonymous objects by rules.

Recall that also in SPARQL, even though *reasoning* allows to derive that Joe's father, *jf*, is a person, and also $\mathit{jf} \in \exists \mathsf{hasFather.Person}$, holds, a query {?X hasFather ?Y} will not yield an answer with ?X/joe (and also not with ?X/*jf*).

When considering such an axiom like $\mathsf{Person} \sqsubseteq \exists\ \mathsf{hasFather.Person}$ as a rule, using a function symbol for object invention as usual in F-Logic,

```
father(X):c_Person :- X:c_Person.
```

would create an infinite chain of objects father(father(...(joe))).

From that, the following strategy is a applied for such objects:

Strategy 1 *Don't derive too much about objects that are not explicitly named. Don't use function symbols in rule heads.*

If only **positive** rules are used, missing existential objects can lead to missing data, but not to wrong derivations. E.g. the rule

```
X:c_Uncle :- X:c_Person, X[hasSibling→→Y], Y[hasChild→→Z].
```

with the facts {Person(joe), hasSibling(joe,mary), Parent(mary)}, the latter equivalent with ∃hasChild.Person(mary), would not be sufficient to derive that Joe is an uncle.
If **negative** rules are considered, we have to care for rules like

```
X:c_Childless :- X:c_Person, not X[hasChild→→_Y].
X:c_Fatherless :- X:c_Person, not X[hasFather→→_Y].
```

The first rule would derive c_Childless(X) for all persons, including parents p for whom no *explicit* filler for p.hasChild is known, and the second will derive c_Fatherless(X) for all "bordering" persons of the ontology whose father is only implicitly known.

Strategy 2 *Export all relevant implicitly known "border" objects with their derived properties from the existential axiom from Jena to F-Logic. Then, derive only relevant information about them.*

Which such implicitly known "border" objects are relevant? A border object is relevant, if it is concerned by an atom in some rule. This can be by done inspecting the graph of each rule body (note that e.g.,

```
c_Grandfatherless(X) :- X:c_Person, hasFather(X,_Y), not hasFather(Y,_Z).
```

makes even the grandfathers relevant border objects). Such implicit border objects must/need only be exported if there is no explicit filler for the corresponding role.

Border objects are marked as such, being members of an internal class borderobject. When they occur in a rule head, only relevant properties are actually derived (again based on the graphs of the rules).

Note that this idea is similar to the one presented in [Luk07] that is based on the Herbrand base/model.

3.6 Practical Issues and Additional Functionality for Daily Life

On one hand, the handling of anonymous objects is still "critical", especially in the context of negation. On the other hand, a careful design of the ontology and the rules allow for a reasonable expressiveness obtained by declarative formalisms, which otherwise must be implemented procedurally (and then has no logical semantics as all). As one of the aims of the approach is to provide a working architecture for individual nodes in the Semantic Web, we went for a pragmatic realization. Some additional functionality that is useful for daily life comes with the use of Florid:

– built-in predicates and operations on strings and numbers,
– creation of URIs (as they are basically also strings – only the membership atom *s*:url distinguishes them from simple strings).

Example 3 (Train Connections). Consider a train database which contains the "atomic" connections. Assume that for the urls of cities and connections, a globally agreed structure is used. Consider the following fragment (in N3):

```
<travel://db/connections/Hannover-Goettingen> a <travel:meta#Connection>;
  <travel:meta#from> <geo://de/Hannover>; <travel:meta#to> <geo://de/Goettingen>;
  <travel:meta#distance> 120.
<travel://db/connections/Goettingen-Kassel> a <travel:meta#Connection>;
  <travel:meta#from> <geo://de/Goettingen>; <travel:meta#to> <geo://de/Kassel>;
  <travel:meta#distance> 60.
```

and a (simplified) rule that computes composite connections (in F-Logic syntax, where URLs are treated by strings *s*:url):

```
(U:"travel://meta#Connection")["travel:meta#from"→X; "travel:meta#to"→Z;
        " travel:meta#distance"→D], U:url
:-  (C1:"travel://meta#Connection")["travel:meta#from"→X; "travel:meta#to"→Y;
        " travel:meta#distance"→D1],
    (C2:"travel://meta#Connection")["travel:meta#from"→Y; "travel:meta#to"→Z;
        "travel:meta#distance"→D2],
    U = "travel://db/connections/" + X.name + "-" + Y.name, D = D1 + D2.
```

which will generate (already mapped back to N3)

```
<travel://db/connections/Hannover-Kassel> a <travel:meta#connection>;
  <travel:meta#from> <geo://de/Hannover>; <travel:meta#to> <geo://de/Kassel>;
  <travel:meta#distance> 180.
```

Additionally, Florid allows for parsing of HTML and XML data, which can then be transformed by F-Logic rules into RDF to be added to the ontology.

4 Defaults

4.1 Inheritance in F-Logic

Inheritance atoms in F-logic have the form $c[p \bullet\!\!\to v]$: the class c provides the *inheritable scalar* property p with default value v. For a member $o : c$ without any intermediate class, inheritance results in $o[p \to v]$; for a subclass $d :: c$, inheritance results in $d[m{\bullet\!\!\rightarrow}v]$; analogously for multivalued $c[p{\bullet\!\!\twoheadrightarrow}v]$. Inheritance of defaults is intended to take place, if "nothing else is known".

In [KLW95], *inheritance-canonic* models are defined, based on *inheritance triggers* which extend the above fixpoint semantics: default inheritance is applied after the minimal/stratified model is computed and the rules do not derive any more facts. Objects where for an inheritable property, no value has been derived so far inherit the default, and the minimal/perfect model computation is applied again. Although this definition is formulated in a rather procedural way, we have shown in [MK01] that this semantics is "reasonable" and in most cases compatible with the Default Logic [Rei80,Poo94] semantics. Only when application of rules after applying a default results in an inconsistency, or "attacks" the applicability of the default, a non-supported (wrt. the Default Logic semantics) model results. Below, we show that even this effect is avoided by the DL+Florid architecture where F-Logic reasoning is applied as a "slave" whose results can further be controlled.

4.2 Nonmonotonic Inheritance by Default Logic

In *Default Logic* [Rei80,Poo94], defeasible reasoning is expressed by *defaults*: a default $d = \dfrac{\alpha : \beta}{w}$ consists of a *precondition* α, a *justification* β and a *consequence* w. Given α, if β can be assumed consistently, one can conclude w. A *default theory* is a pair $\Delta = (D, F)$ where D is a set of defaults and F is a set of formulas.

In an inheritance framework, the superclass condition belongs to α; whereas the checks that inheritance is not preempted by an intermediate class and that the inherited value must be consistent with the knowledge (wrt. the logical rules of the program) fall under β. For characterizing inheritance, only a specialized form of defaults is needed, called *semi-normal defaults* where the precondition $\alpha(\bar{x})$ is a conjunction of atoms, the consequence $w(\bar{x})$ is also an atomic formula, and $\forall\bar{x} : \beta(\bar{x}) \rightarrow w(\bar{x})$ holds. Translating the path-based concept of *inheritance networks*, including avoidance of *decoupling* inheritance in F-Logic syntax can be specified by defaults of the form (cf. [MK01])

$$D_{inh} = \frac{\begin{array}{l} O : C\ ,\ C[M{\bullet\!\!\rightarrow}V]\ : \\ \quad \forall C'((O : C' \wedge C' :: C) \rightarrow C'[M{\bullet\!\!\rightarrow}V])\ ,\ O[M{\rightarrow}V] \end{array}}{O[M{\rightarrow}V]}\ .$$

The semantics of a default theory is defined in terms of *extensions*. A theory $\mathcal{T}$ is an extension of $\Delta = (D, F)$ if it satisfies certain requirements [Rei80,Poo94,Mak94]; the definitions are non-constructive (where a *quasi-inductive* definition at least gives a guess-and-check characterization).

In our case, the consequent of a default is always a single ground fact. Thus, it is again sufficient to define and analyze the semantics only with the underlying Herbrand Structures, not with theory reasoning.

4.3 Handling Nonmonotonic Inheritance

In the same way as negation, the default reasoning conflicts with the OWA. Here, the conflict is intended: the main reason behind default inheritance is to be able to find a reasonable set of *beliefs* in a situation of incomplete knowledge when any "safe" reasoning –both OWL OWA reasoning and rule-based CWA reasoning– does not lead to additional knowledge. Thus, default inheritance atoms are evaluated in an *outer* iteration. When an alternating fixpoint of OWL and rules is reached, apply one applicable default, and restart the alternating fixpoint.

In our setting, this is accomplished by saving the current model, applying the default, applying OWL and rules reasoning up to the next fixpoint, and if this structure is consistent, continue with it. If a theory has extensions, then this process leads to an extension; otherwise it at least results in a non-extensible structure that is consistent, but contains "non-fired" default instances. Note that the possibility to put the intermediate model aside during the computation controlled by the Jena module allows to accomplish this guarantee in contrast to the F-Logic/Florid-only semantics discussed in [MK01].

In [BH95], it is shown that even for the simple DL $\mathcal{ALCF}$, Reiter's semantics for open defaults leads to undecidability. When considering only individuals that are mentioned explicitly in the ABox, the task becomes decidable. In our approach, the ontology is restricted to explicit facts before submitting it to Florid – thus, the above obviously applies. Note that implicit border objects have to be ignored when checking for applicable defaults.

5 Comparison

While both OWL-DL and function-free Horn rules are decidable fragments of first-order logic, however, the combination leads to undecidability in general. *DLP* [GHVD03] is the, not very expressive, but decidable, intersection of DL and LP. The other end of the spectrum is *SWRL* (earlier: ORL) [HPS04] which is the *union* of DL and LP, which is in general undecidable.

$\mathcal{AL}$-log [DLNS91] proposes hybrid rules in a constraint LP style, where the LP Datalog clauses in the body are extended with DL class membership constraints; the heads of the rules may only contain the LP predicates. CARIN [LR96] extends this to allow also DL roles in the Datalog bodies. *Role-safeness*, i.e., in every DL role atom, at least one variable also occurs in a base predicate (i.e., an LP predicate which does not occur in any rule head), guarantees decidability (wrt. $\mathcal{ALCNR}$). A similar strategy is followed in *DL-safe* rules [MSS05], where it is shown that it is sufficient to require *each* variable in the rule to occur in a non-DL-atom in the rule body (wrt. the more expressive $\mathcal{SHOIN}(D)$/OWL-DL); but here DL atoms are also allowed to occur in rule heads. *DL-safety* also makes sure that each variable is bound only to individuals that are *explicitly* known in the ABox.

The DL+log [Ros06] approach provides a tighter integration and shows that a "weak safeness" condition, i.e., every *head* variable must occur in an LP atom in the body, guarantees decidability. Thus, in contrast to previous approaches, it is allowed to have variables in the body that only appear in DL atoms (thus, the language now covers conjunctive queries over DL). Again, the focus is that the individuals for which something is derived are *explicitly* known (while implicit objects can be significant in the body). Decidability is obtained for all DLs where CQ/UCQ containment is decidable; this includes the logic $\mathcal{DLR}$ [CGL+98], which is weaker than e.g. OWL-DL.

In [MR07], the *MKNF (Minimal Knowledge and Negation as Failure)* [Lif91] idea is applied to DL & LP to obtain a unifying framework which covers e.g. DL+log, SWRL, LP under stable models semantics, and Default Logic with fixed universe. The resulting logic is based on the modal logic S5 and preferential models. Again, *DL-safety* guarantees decidability. The special interaction of open- and closed-world reasoning (DL predicates can also occur under CWA negation) allows to express things that cannot be expressed in other approaches.

There are also some "loose integration" approaches, in the sense that the rules part contains queries to the ontology, again in the style of constraint LP. These do not derive ontology predicates in the head: In DLV/DLVhex (DLV with higher-order and external atoms) [ELST04,EIST06], the DL atoms are "external" to the DLV framework (disjunctive LP) that is based on Answer Set Programming. In a similar way, [DM07] integrates DL atoms into an LP framework (normal logic programs under well-founded semantics).

In contrast, OWL Flight [dBLPF05], reduces the OWL part to what can be expressed with (F-Logic) rules, under CWA, and adds constraints.

In [Luk07], the perspective is changed: the above approaches consider the DL, theory reasoning, perspective. When changing to the perspective of rule-based systems, considering Herbrand structures as a base yields decidability without any syntactic restrictions. The paper proposes a guess-and-check algorithm which is in general in $\mathsf{NExp}^{\mathsf{NP}}$, and has polynomial data complexity for normal or stratified programs in combination with DL-Lite. The approach is the most similar to ours amongst the above ones.

Considering implementations, there is the DLVhex system [ELST04,EIST06], the prototype of [DM07], the upcoming SWRL [HPS04,HPSB+04] support in Pellet [Pel], and KAON2 [KA] which encodes OWL into disjunctive Datalog and allows for DL-safe rules.

6 Conclusion

We have described the combination of (i) OWL-DL, (ii) F-Logic rules, and (iii) default inheritance, consisting of a Jena-based OWL node that employs Florid as a "slave" for evaluating rules. In case that one of the components is empty, the semantics coincides with the usual semantics of the respective formalism:

– If the OWL-DL part of the ontology is empty (or a simple schema+data style database which does not contribute to the reasoning), the resulting system just applies the F-Logic rules and default inheritance in a controlled way, resulting in a safe semantics wrt. the investigations in [MK01].

– If the F-Logic rule part is empty, the system just implements OWL-DL with default inheritance. Again, inheritance is controlled in such a way that it inherits only when the application is consistent, exactly mirroring the semantics from Default Logic given in [Rei80].
– If the predicates that occur in rule heads are disjoint from those of the OWL ontology (i.e., the rule part only queries the ontology, as in [ELST04,EIST06] and [DM07]), the semantics is correct and complete for stratified negation.
– If the predicates that occur in rule *bodies* are disjoint from those of the OWL ontology (i.e., the rule part serves for populating the ontology, e.g., by information extraction from Web pages), the semantics is also correct and complete for stratified negation.

The approach is under implementation in the DL+Florid prototype[1]. It provides a feasible integration of OWL ontologies and rules that is to be seen as a declarative alternative to approaches that implement the same by pure (Java) programming. Its advantages are the declarative specification of the rules that allow for rapid prototyping and flexible adaptation of the rules.

Ongoing and future work is concerned with implementing the inheritance semantics completely and investigating the possibilities to enhance the handling of existential anonymous objects. We expect that syntactical analysis of the rules and the dependency graph in many cases allows for identifying a finite set of anonymous objects and their properties that is sufficient for guaranteeing correctness and completeness for all derived facts about explicitly known objects.

Acknowledgements. This research has been funded by the European Commission within the 6th Framework Programme project REWERSE, no. 506779.

References

[BCM+03] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook.* Cambridge University Press, 2003.
[BH95] F. Baader and B. Hollunder. Embedding Defaults into Terminological Knowledge Representation Formalisms. *J. of Automated Reasoning*, 14:149–180, 1995.
[CGL+98] D. Calvanese, G. D. Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Description Logic Framework for Information Integration. In *Principles of Knowledge Representation and Reasoning (KR)*, pp. 2–13, 1998.
[dBLPF05] J. de Bruijn, R. Lara, A. Polleres, D. Fensel. OWL DL vs. OWL Flight: conceptual modeling and reasoning for the semantic Web. In *WWW Conf.*, 2005.
[DLNS91] F. M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. A Hybrid System with Datalog and Concept Languages. In *Trends in Artificial Intelligence; AI*IA'91*, Springer LNCS 549, pp. 88–97, 1991.
[DLNS98] F. M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. AL-log: Integrating Datalog and Description Logics. *J. Intell. Inf. Syst.*, 10(3):227–252, 1998.
[DM07] W. Drabent and J. Małuszyński. Well-Founded Semantics for Hybrid Rules. In *Web Reasoning and Rule Systems (RR)*, Springer LNCS 4524, pp. 1–15, 2007.

[1] `http://www.semwebtech.org/DLFlorid`

[EIST06] T. Eiter, G. Ianni, R. Schindlauer, and H. Tompits. Effective Integration of Declarative Rules with External Evaluations for Semantic-Web Reasoning. In *Europ. Semantic Web Conf. (ESWC)*, pp. 273–287, 2006.

[ELST04] T. Eiter, T. Lukasiewicz, R. Schindlauer, and H. Tompits. Combining Answer Set Programming with Description Logics for the Semantic Web. In *Principles of Knowledge Representation and Reasoning (KR)*, pp. 141–151, 2004.

[FHK$^+$97] J. Frohn, R. Himmeröder, P.-T. Kandzia, G. Lausen, and C. Schlepphorst. FLORID: A Prototype for F–Logic. *Intl. Conf. on Data Engineering (ICDE)*, 1997.

[FLO06] Florid Homepage. http://www.informatik.uni-freiburg.de/~dbis/florid/, 2006.

[GHR94] D. M. Gabbay, C. J. Hogger, and J. A. Robinson. *Handbook of Logic in Artificial Intelligence and Logic Programming.* Oxford Science Publications, 1994.

[GHVD03] B. Grosof, I. Horrocks, R. Volz, and S. Decker. Description logic programs: combining logic programs with description logic. In *WWW Conf.*, pp. 48–57, 2003.

[HKS06] I. Horrocks, O. Kutz, and U. Sattler. The Even More Irresistible SROIQ. In *Principles of Knowledge Representation and Reasoning (KR)*, pp. 57–67, 2006.

[HPS04] I. Horrocks and P. Patel-Schneider. A Proposal for an OWL Rules Language. In *WWW Conf.*, pp. 723–732, 2004.

[HPSB$^+$04] I. Horrocks, P. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. http://www.w3.org/Submission/SWRL/, 2004.

[HS05] I. Horrocks and U. Sattler. A Tableau Decision Procedure for SHOIQ(D). In *Intl. Joint Conf. on Artificial Intelligence (IJCAI)*, 2005.

[Jen] Jena: A Java Framework for Semantic Web Appl's. http://jena.sourceforge.net.

[KA] KAON2 – Ontology Management for the Semantic Web. http://kaon2.semanticweb.org/.

[KLW95] M. Kifer, G. Lausen, and J. Wu. Logical Foundations of Object-Oriented and Frame-Based Languages. *Journal of the ACM*, 42(4):741–843, 1995.

[Lif91] V. Lifschitz. Nonmonotonic Databases and Epistemic Queries. In *IJCAI*, 1991.

[LR96] A. Y. Levy and M.-C. Rousset. CARIN: A Representation Language Combining Horn Rules and Description Logics. In ECAI, pp. 328–334, 1996.

[Luk07] T. Lukasiewicz. A Novel Combination of Answer Set Programming with Description Logics for the Semantic Web. In *ESWC*, 2007.

[Mak94] D. Makinson. General Patterns in Nonmonotonic Reasoning. In [GHR94].

[May07] W. May. The MONDIAL Database, 1999–2007. http://dbis.informatik.uni-goettingen.de/Mondial/.

[MK01] W. May and P.-T. Kandzia. Nonmonotonic Inheritance in Object-Oriented Deductive Database Languages. *J. of Logic and Computation*, 11(4), 2001.

[MR07] B. Motik and R. Rosati. A Faithful Integration of Description Logics with Logic Programming. In *Intl. Joint Conf. on Artificial Intelligence (IJCAI)*, 2007.

[MSS05] B. Motik, U. Sattler, and R. Studer. Query Answering for OWL-DL with rules. *J. of Web Semantics*, 3(1):41–60, 2005.

[MSvL06] W. May, F. Schenk, and E. von Lienen. Extending an OWL Web Node with Reactive Behavior. In *Principles and Practice of Semantic Web Reasoning (PPSWR)*, Springer LNCS 4187, pp. 134–148, 2006.

[OWL04] OWL Web Ontology Language. http://www.w3.org/TR/owl-features/, 2004.

[Pel] Pellet: An OWL DL Reasoner. http://pellet.owldl.com.

[Poo94] D. Poole. Default Logic. In [GHR94].

[Rei80] R. Reiter. A Logic for Default Reasoning. *Artificial Intelligence*, 12(1,2), 1980.

[Ros06] R. Rosati. DL+log: Tight Integration of Description Logics and Disjunctive Datalog. In *Principles of Knowledge Representation and Reasoning (KR)*, 2006.

[SPQ06] SPARQL Query Language for RDF. www.w3.org/TR/rdf-sparql-query/, 2006.