Formalizing Object-ontological Mapping Using F-logic

Martin Ledvinka[1] and Petr Křemen[1]

Department of Computer Science, Faculty of Electrical Engineering,
Czech Technical University in Prague, Prague 6 - Dejvice, Czech Republic
{martin.ledvinka, petr.kremen}@fel.cvut.cz

Abstract. Ontologies can represent a significant asset of domain-specific information systems, written predominantly using the object-oriented paradigm. However, to be able to work with ontological data in this paradigm, a mapping must ensure transformation between the ontology and the object world. While many software libraries provide such a mapping, they lack standardization or formal guarantees of its semantics. In this paper, we provide a formalism for mapping ontologies between description logics and F-logic, a formal language for representing object-oriented programming languages. This formalism allows to precisely specify the semantics of the object-ontological mapping and thus ensure a predictable shape and behavior of the object model.

Keywords: Object-ontological Mapping · $\mathcal{SROIQ}$ · F-logic.

1 Introduction

The object-oriented paradigm (OOP) has been a dominant software development technique in the past two decades, mainly due to its ability to represent the underlying domains in a natural and understandable way [4]. Ontologies, on the other hand, can significantly increase the capabilities of information systems, especially due to their formal semantics (in this paper, we consider description logic (DL) [2] as the language backing the formal semantics of ontologies), elements with shared meaning and global identification, and inference enabled by expressive languages. Yet, to be able to fully embrace the benefits of ontologies in OOP, a *mapping* is needed to transform data between the two worlds. Many software libraries provide such functionality, however, without sufficient guarantees as to the semantics of the mapping.

One differentiating aspect between an object model and a DL ontology[1] is the *open world assumption* of the latter – DL ontologies assume incomplete knowledge of the domain. However, most domain-specific information systems assume data completeness and thus, if a fact cannot be derived from the existing

[1] In the sequel, we consider mapping of DL ontology specifications since the most widespread ontology-related standards (e.g., OWL) and relevant tools are based on the description logic formalism.

data, it is considered false. To overcome this mismatch, *integrity constraints* can be used to place restrictions on the knowledge base. Consider a simple vocabulary management system which needs to keep track of authors of the vocabularies. In description logics, this is represented by placing existential quantification on the vocabulary records. Unfortunately, this merely ensures that *some* author exists for each vocabulary. But they may remain unknown which is hardly sufficient for ensuring integrity of data created by the system. Integrity constraints can be used to enforce that an author is explicitly assigned to each record.

Thus, the goal of this paper is to provide a formalism for *object-ontological mapping* (OOM) moderated by integrity constraints. We chose F-logic as a vessel for this formalism. The main reasons are that it is a logic-based language, it has been used to describe ontologies and it is specifically designed to represent the structural aspects of object-oriented languages. Its syntax allows to concisely represent the most common constructs needed by object-oriented domain models – class hierarchies, local restrictions on property value types, possible cardinality restrictions and individual assertions.

1.1 Running Example

We shall use the following example DL ontology throughout this paper to illustrate the mapping. $\mathcal{T}$ represents the ontology schema, $\mathcal{A}$ are the actual data and $\mathcal{IC}$ are *integrity constraints* placed on the ontology. All the corresponding notions will be explained in Section 2.

In the example, we declare an asset and specify that it has to have an author and it may have a last editor. This generic ontology is restricted by integrity constraints for a system working with vocabularies, which are kinds of assets. The constraints specify the same cardinalities of both author and last editor as $\mathcal{T}$, but require their values to be users of the system.

$$\begin{aligned}\mathcal{T} =& \{Asset \sqsubseteq = 1 author.\top, Asset \sqsubseteq \leqslant 1 lastEditor.\top,\\ & Vocabulary \sqsubseteq Asset, author \sqsubseteq editor, lastEditor \sqsubseteq editor\}\\ \mathcal{A} =& \{User(Tom), User(Sarah), Vocabulary(MetropolitanPlan)\}\\ \mathcal{IC} =& \{Vocabulary \sqsubseteq \forall author.User, Vocabulary \sqsubseteq = 1 author.User,\\ & Vocabulary \sqsubseteq \forall lastEditor.User, Vocabulary \sqsubseteq \leqslant 1 lastEditor.User\}\end{aligned}$$

The paper is structured as follows: Section 2 provides the necessary theoretical background, Section 3 presents the object-ontological mapping, while Section 4 introduces the mapping of integrity constraints. Section 5 discusses related work and Section 6 concludes the paper.

2 Background

This section presents the most important notions of the description logic $\mathcal{SROIQ}$, application of *integrity constraints* to ontologies, and F-logic.

2.1 $\mathcal{SROIQ}$

$\mathcal{SROIQ}$ [15] is an expressive description logic (DL), i.e., a decidable sub-language of the first order logic (FOL), used to describe ontologies. Each $\mathcal{SROIQ}$ ontology $\mathcal{O}$ is comprised of a *terminology* (TBox and RBox), which describes the schema of the ontology, and a set of individual assertions representing actual data (ABox)[2]. TBox consists of a *concept hierarchy* where concepts can be either *atomic* or *concept descriptions* of the following forms: $\neg C$, $C \sqcap D$, $C \sqcup D$, $\geqslant nR.C$, $\leqslant nR.C$, $\exists R.Self$, $\{a\}$, $\forall R.C$, $\exists R.C$, where C, D are concepts, R is a role, n is a non-negative integer and a is an individual. RBox consists of a hierarchy of roles and axioms stating their properties, for instance, $Sym(R)$ denoting a symmetric role, or $Dis(R, Q)$ denoting disjoint roles. The schema also contains built-in concepts $\top$, $\bot$ and a built-in universal role R_U.

Individual assertions are of the form $C(a)$, $R(a, b)$, $a = b$ and $a \neq b$, where a and b are individuals, C is a concept and R is a role. The set N_C represents concept names, N_R role names, and N_I denotes the set of individual names.

The semantics of a $\mathcal{SROIQ}$ ontology $\mathcal{O}$ is given by an *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is the *domain* of the interpretation and $\cdot^{\mathcal{I}}$ is the *interpretation function*. This function assigns to every atomic concept A a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, to every atomic role R a binary relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ and to every individual an element of $\Delta^{\mathcal{I}}$. $\top^{\mathcal{I}}$ is $\Delta^{\mathcal{I}}$, $\bot^{\mathcal{I}}$ is the empty set $\emptyset$ and $R_U^{\mathcal{I}}$ is $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. $\mathcal{I}$ is a model of an ontology $\mathcal{O}$ consisting of a TBox $\mathcal{T}$, an RBox $\mathcal{R}$, and an ABox $\mathcal{A}$ ($\mathcal{I} \models \mathcal{O} = \mathcal{T} \cup \mathcal{R} \cup \mathcal{A}$) if it satisfies all the axioms in $\mathcal{O}$. A set of axioms Θ *logically entails* an axiom θ ($\Theta \models \theta$) if and only if all models of Θ are also models of θ. Concrete rules for interpretation of concept descriptions and axioms are described in [15] and we omit them here for the lack of space.

2.2 Integrity Constraints

The intention of integrity constraints (ICs) in the area of application access to DL ontologies is mostly to restrict the open-world nature of (a portion of) an ontology. While OWL does allow to express certain constraints, its expressiveness in this regard is limited. For example, a DL ontology might require that every asset must have a unique author ($Asset \sqsubseteq =1 author.\top$). If an asset v does not have one, the reasoner will infer an anonymous individual, which will satisfy the requirement for the sake of ontology consistency (i.e., existence of a model). However, a vocabulary management system requires a stronger condition to be satisfied – every vocabulary (a special kind of asset) must have a *known* author of type user. Such a constraint cannot be enforced in OWL with standard semantics. However, there are approaches which allow this type of restrictions. [10] introduces *Minimal Knowledge and Negation as Failure* logics, while [24] uses minimal Herbrand models. Unfortunately, both of these can lead to counter-intuitive results, as was pointed out in [28]. It presents a novel approach which

[2] $\mathcal{SROIQ}$ allows expressing individual assertions using TBox axioms with nominals. However, ABox assertions provide a natural, easy to read syntax which we will use throughout this paper.

remedies these issues. A more recent effort in [26] discusses the flaws of all of the aforementioned solutions and proposes the use of DBox-based *completely specified* concepts and roles. However, not even this approach is immune to debatable results. Consider the following example:

$$\begin{aligned}\mathcal{T} =&\{Employee \sqsubseteq Person, Flight \sqsubseteq \exists hasPassenger.Person\}\\ \mathcal{A} =&\{Flight(c), Flight(d)\}\\ \mathcal{DB} =&\{Person(a), Employee(b), hasPassenger(c,a), hasPassenger(d,b)\}\end{aligned}$$

We put *hasPassenger*, *Person* and *Employee* into the DBox, so that no unexpected instances are generated. However, this will cause an IC violation, because $Person(b)$ will be inferred for a completely specified concept *Person*. The approach of Tao et al. [28] does not suffer from such issues, because, while it does work only with named individuals, it does not prevent inference of new types/relationships for them. We will be using it for this work as we consider it the most suitable for the OOM case.

Integrity Constraint Semantics Since it highly relevant for our work, let us provide a brief overview of the integrity constraint semantics defined by Tao et al. (details can be found in [28]). They define an *IC interpretation* as a pair $\mathcal{I},\mathcal{U}$ where $\mathcal{I}$ is a $\mathcal{SROIQ}$ interpretation defined over $\Delta^{\mathcal{I}}$ and $\mathcal{U}$ is a set of $\mathcal{SROIQ}$ interpretations. An IC interpretation of a concept C and a role R is then defined as follows:

$$\begin{aligned}C^{\mathcal{I},\mathcal{U}} =&\{x^{\mathcal{I}} \mid x \in N_I \ s.t.\ \forall \mathcal{J} \in \mathcal{U}, x^{\mathcal{J}} \in C^{\mathcal{J}}\}\\ R^{\mathcal{I},\mathcal{U}} =&\{\langle x^{\mathcal{I}}, y^{\mathcal{I}}\rangle \mid x,y \in N_I \ s.t.\ \forall \mathcal{J} \in \mathcal{U}, \langle x^{\mathcal{J}}, y^{\mathcal{J}}\rangle \in R^{\mathcal{J}}\}\end{aligned}$$

IC interpretation $\mathcal{I},\mathcal{U}$ extends to complex concept descriptions as one would expect, e.g., for $(C \sqcap D)^{\mathcal{I},\mathcal{U}} = C^{\mathcal{I},\mathcal{U}} \cap D^{\mathcal{I},\mathcal{U}}$. This in essence means that a *known* individual a belongs to a concept C under the IC interpretation $\mathcal{I},\mathcal{U}$ if it belongs to it in all interpretations in $\mathcal{U}$. Similar principles apply to axiom satisfaction, e.g., $C \sqsubseteq D$ is satisfied in $\mathcal{I},\mathcal{U}$ under the condition that $C^{\mathcal{I},\mathcal{U}} \subseteq D^{\mathcal{I},\mathcal{U}}$. Tao et al. also introduce the notion of *minimal equality models*. Our approach uses an analogous construct, therefore, we postpone its discussion to Section 4.

2.3 F-logic

F-logic [18,17,1] is a formalism rooted in FOL which can be used to describe structural aspects of object-oriented or frame-based languages. It has model-theoretic semantics and a sound and complete proof theory. In the discussion of F-logic syntax, we use the revised version of [1] and w.l.o.g. omit the distinction between inheritable and non-inheritable methods. Given that F-logic allows one to specify relatively complex programs, we use a restricted variant of F-logic, which is suitable for mapping of DL, but does not contain, for instance, methods with arbitrary arity (we are using only *attributes* – parameterless methods). We use *sorted* F-logic, so that (atomic) classes are disjoint from individuals and methods (much like classes, individuals and properties are disjoint in DL).

F-logic Syntax The alphabet of an F-logic language $\mathcal{L}$ consists of

- A set of *object constructors* $\mathcal{F} = \mathcal{C} \cup \mathcal{R} \cup \mathcal{E} \cup \mathcal{A}$, where $\mathcal{C}$ is a set of class names (0-ary function symbols), $\mathcal{R}$ is a set of methods (0-ary function symbols), $\mathcal{E}$ is a set of instances (0-ary function symbols), and $\mathcal{A}$ is a set of function symbols (it essentially allows us to *parameterize* concept constructors, as will be seen in Section 3). $\mathcal{C}$, $\mathcal{R}$, $\mathcal{E}$, and $\mathcal{A}$ are mutually disjoint,
- A set of predicate symbols $\mathcal{P}$,
- An infinite set of variables $\mathcal{V}$,
- Auxiliary symbols like (,), [,], $\rightarrow$ etc.,
- Logical connectives and quantifiers $\wedge$, $\vee$, $\neg$, $\forall$, $\exists$.

An *id-term* is a first-order term composed of an object constructor and variables. A variable-free object constructor is called a *ground id-term* and the set of all ground id-terms is denoted $U(\mathcal{F})$. Formulas in F-logic can be either *molecular formulas*, or complex formulas consisting of other formulas connected using logical connectives and quantifiers. Molecular formulas can be:

1. *Is-a* assertions of the form $A :: B$ or $o : A$, where o, A, B are id-terms,
2. *Object molecules* of the form O[a ';' separated list of method expressions]. Where method expressions can be:

- *data expressions* of the form $m \rightarrow v$, where m and v are id-terms (v is the attribute value),
- *Signature expressions* of the form $m \Rightarrow (T_1, ..., T_n)$, where $n \geq 1$ and m and T_i are id-terms (T_i are the return types).

In short, data expressions represent attribute values, whereas signature expressions represent their return types.

F-logic Semantics Semantics of F-logic is specified using *F-structures*. Before we define an F-structure, we need several additional notions.

For a pair of sets U, V, $Total(U, V)$ denotes the set of all *total* functions from U to V. Similarly, $Partial(U, V)$ denotes the set of all *partial* functions from U to V. We use $\mathcal{P}(U)$ to express the power set of U. $\mathcal{P}_\uparrow(U)$ is the set of all *upward-closed* subsets of U. A set $V \subseteq U$ is upward closed if for $v \in V$, $u \in U$, $v \prec_U u$ implies $u \in V$, where $\prec_U$ is an irreflexive partial order on U (see below). $PartialAM_{\prec_U}(U, \mathcal{P}_\uparrow(U))$ denotes the set of all partial *anti-monotonic* functions from U to $\mathcal{P}_\uparrow(U)$. A function f is *partial anti-monotonic* if for vectors $\boldsymbol{u}, \boldsymbol{v} \in U^k$, $\boldsymbol{v} \prec_U \boldsymbol{u}$, if $f(\boldsymbol{u})$ is defined, then $f(\boldsymbol{v})$ is also defined and $f(\boldsymbol{u}) \subseteq f(\boldsymbol{v})$.

An F-structure is then a tuple $\mathbf{I} = \langle U, \prec_U, \in_U, I_F, I_\mathcal{P}, I_\rightarrow, I_\Rightarrow \rangle$, where:

- U is the domain of $\mathbf{I}$ consisting of disjoint subdomains $U_\mathcal{E}$, $U_\mathcal{C}$, $U_\mathcal{R}$, $U_\mathcal{A}$,

- $\prec_U$ is an irreflexive partial order on $U_{\mathcal{C}\cup\mathcal{A}}$ representing the subclass relationship,[3]
- $\in_U$ is a binary relationship on $U_{\mathcal{E}} \times U_{\mathcal{C}\cup\mathcal{A}}$ specifying instance membership in classes,
- $I_F : \mathcal{F} \to \bigcup_{k=0}^{\infty} Total(U^k, U)$ is a mapping which represents function symbols from $\mathcal{F}$ by functions from U^k to U. For $k = 0$, $I_F(f)$ can be identified with an element of U. I_F maps names to their respective subdomains, e.g., class names from $\mathcal{C}$ to $U_{\mathcal{C}}$,
- $I_{\mathcal{P}}(p) \subseteq U^n$ for any n-ary predicate symbol $p \in \mathcal{P}$,
- $I_{\rightarrow} : U_{\mathcal{R}} \to Partial(U_{\mathcal{E}}, \mathcal{P}(U_{\mathcal{E}}))$,
- $I_{\Rightarrow} : U_{\mathcal{R}} \to PartialAM_{\prec_U}(U_{\mathcal{C}\cup\mathcal{A}\cup\mathcal{E}}, \mathcal{P}_{\uparrow}(U_{\mathcal{C}\cup\mathcal{A}}))$.

Remarks The use of *upward-closed* sets is important for class hierarchies – it means that along with each class, the set also contains all its superclasses. The relationship between $I_{\rightarrow}$ and $I_{\Rightarrow}$ is such that $I_{\Rightarrow}$ defines the target (range) type of an attribute, whereas $I_{\rightarrow}$ defines particular values.

A variable assignment ν is a mapping from the set of variables, $\mathcal{V}$, to the domain U, which extends to id-terms as follows: $\nu(d) = I_{\mathcal{F}}(d)$ if $d \in \mathcal{F}$ has arity 0 and $\nu(f(..., t, ...)) = I_{\mathcal{F}}(f)(..., \nu(t), ...)$. Intuitively, given an F-structure $\mathbf{I}$ and a variable assignment ν, a molecule $t[...]$ is *true* under $\mathbf{I}$ w.r.t. to ν, written $\mathbf{I} \vDash_{\nu} t[...]$, if and only if the object $\nu(t)$ has the properties defined by the F-molecule. For example, $\mathbf{I} \vDash_{\nu} (O :: P)$ iff $\nu(O) \preceq_U \nu(P)$. For attributes, this means that there exist functions interpreting them and they have the right return values (types), e.g., $\mathbf{I} \vDash_{\nu} q[m \to v]$ iff $I_{\rightarrow}(\nu(m))(\nu(q))$ is defined and contains $\nu(v)$. An object molecule is considered a conjunction of method expressions. Precise definitions of logical implication in F-logic can be found in [18], Sec. 5.2. Satisfaction of complex formulas (using logical connectives and quantifiers) is defined in the usual first-order sense. An F-logic theory $\mathcal{S}$ logically implies an axiom α ($\mathcal{S} \vDash_{\nu} \alpha$) iff all models of $\mathcal{S}$ are also models of α. Since we will be working with closed formulas only, we can omit the variable assignment identifier. Instead, we shall denote F-logic semantic implication by $\models^F$ to distinguish it from DL entailment. We omit discussion of properties of F-structures here due to the lack of space. Nevertheless, since these properties do affect the formalization, the reader should refer to [18], Sec. 7, if necessary.

Queries An F-logic *query* Q is a molecule. The set of answers to Q w.r.t. a set of formulas P is the smallest set of molecules that:

- contains all instances of Q (variable assignments for all variables in Q) that are found in the model of P,
- is closed under $\models^F$ (see [18], Sec. 12.1.2).

[3] $U_{\mathcal{C}\cup\mathcal{A}}$ is an abbreviation for $U_{\mathcal{C}} \cup U_{\mathcal{A}}$

3 Mapping

We begin this part by introducing the mapping of concept descriptions and ontological axioms. We then show that the mapping preserves entailment in both directions.

The mapping is inspired by [3] but supports a more expressive DL. The use of sorted F-logic and proofs of entailment equivalence are based on [6]. While the version provided here is for $\mathcal{SROIQ}$, the latest version of F-logic supports also datatypes [1], so it could be easily extended to $\mathcal{SROIQ}(\mathcal{D})$. Table 1 shows mapping of concept descriptions. Similar to [3], several new function symbols are introduced – Not, $AtLeast$, $AtMost$, $HasSelf$, Nom, All, $Some \in \mathcal{A}$ – which allow us to represent $\mathcal{SROIQ}$ concept constructs which cannot be directly mapped to F-logic. For instance, $\geqslant nR.C$ does not correspond to $[R_\mathcal{R} \Rightarrow_{\{n:*\}} C_\mathcal{C}]$ because the $\mathcal{SROIQ}$ version admits also R-fillers of other types than C, whereas the F-logic signature expression would require all $R_\mathcal{R}$-fillers to belong to $C_\mathcal{C}$. Also, signature expressions cannot be used to infer the type of values of the corresponding data expressions. For each of the new function symbols, we specify a condition on the underlying F-structures to ensure correct semantics w.r.t. their $\mathcal{SROIQ}$ counterparts.

Table 1. Mapping of concept descriptions. By default, all variables are universally quantified over $\mathcal{E}$. $X_\mathcal{C}$ ($X_\mathcal{R}$) represents a concept (method) name, i.e., a function symbol from $\mathcal{C}$ ($\mathcal{R}$). $AtMost$ is defined analogously to $AtLeast$ and corresponds to $\leqslant nR.C$.

$\mathcal{SROIQ}$	F-logic	F-logic Semantics
A	$A_\mathcal{C}$	
$\neg C$	$Not(C_\mathcal{C})$	$\mathbf{I} \models^F x : Not(C_\mathcal{C})$ iff $I_\mathcal{F}(x) \notin_U I_F(C_\mathcal{C})$
$C \sqcap D$	$C_\mathcal{C}$ and $D_\mathcal{C}$	
$C \sqcup D$	$C_\mathcal{C}$ or $D_\mathcal{C}$	
$\geqslant nR.C$	$AtLeast(n, R_\mathcal{R}, C_\mathcal{C})$	$\mathbf{I} \models^F x : AtLeast(n, R_\mathcal{R}, C_\mathcal{C})$ iff
		$\exists y_1 ... y_n \in U_\mathcal{E}$ s.t. $y_i \in I_\to(I_F(R_\mathcal{R}))(I_F(x))$
		$\wedge y_i \in_U I_F(C_\mathcal{C})$, for $\neg(y_i = y_j)$
$\exists R.Self$	$HasSelf(R_\mathcal{R})$	$\mathbf{I} \models^F x : HasSelf(R_\mathcal{R})$ iff $I_F(x) \in I_\to(I_F(R_\mathcal{R}))(I_F(x))$
$\{a\}$	$Nom(a_\mathcal{E})$	$\mathbf{I} \models^F x : Nom(a_\mathcal{E})$ iff $I_\mathcal{F}(x) = I_\mathcal{F}(a_\mathcal{E})$
$\forall R.C$	$All(R_\mathcal{R}, C_\mathcal{C})$	$\mathbf{I} \models^F x : All(R_\mathcal{R}, C_\mathcal{C})$ iff
		$\forall y \in U_\mathcal{E}$ s.t. $y \in I_\to(I_F(R_\mathcal{R}))(I_F(x)) \Rightarrow y \in_U I_F(C_\mathcal{C})$
$\exists R.C$	$Some(R_\mathcal{R}, C_\mathcal{C})$	$\mathbf{I} \models^F x : Some(R_\mathcal{R}, C_\mathcal{C})$ iff
		$\exists y \in U_\mathcal{E}$ s.t. $y \in I_\to(I_F(R_\mathcal{R}))(I_F(x)) \wedge y \in_U I_F(C_\mathcal{C})$

$\mathcal{SROIQ}$ top (bottom) concept $\top$ ($\bot$) is mapped to F-logic concept $\top_\mathcal{C}$ ($\bot_\mathcal{C}$) for which it must hold $\forall x \in U_\mathcal{E}$, $x \in_U I_F(\top_\mathcal{C})$ ($\forall x \in U_\mathcal{E}$, $x \notin_U I_F(\bot_\mathcal{C})$). Similarly, $\mathcal{SROIQ}$ universal role R_U is mapped to an F-logic method $M_\mathcal{R}$ such that $\forall x, y \in U_\mathcal{E}$ $y \in I_\to(I_F(M_\mathcal{R}))(x)$.

TBox and RBox axiom mapping is shown in Table 2. We make use of F-logic predicates and define conditions under which they are true.

ABox individual assertions are mapped straightforwardly, $C(a)$ as an is-a assertion $a_{\mathcal{E}} : C_{\mathcal{C}}$, $R(a,b)$ as a data expression $a_{\mathcal{E}}[R_{\mathcal{R}} \rightarrow b_{\mathcal{E}}]$ and (in)equality $a = b$ ($a \neq b$) as $a_{\mathcal{E}} = b_{\mathcal{E}}$ ($\neg(a_{\mathcal{E}} = b_{\mathcal{E}})$).

Table 2. Mapping of *TBox* and *RBox* axioms. *RBox* axioms are mapped to predicates, for which we provide satisfaction conditions on the F-structure **I**. $\Rightarrow$ outside of an F-molecule represents regular logical implication. Variables are universally quantified over $U_{\mathcal{E}}$.

$\mathcal{SROIQ}$	F-logic	Condition on **I**
$C \sqsubseteq D$	$C_{\mathcal{C}} :: D_{\mathcal{D}}$	$I_F(C_{\mathcal{C}}) \preceq_U I_F(D_{\mathcal{C}})$
$R \sqsubseteq S$	$subPropertyOf_{\mathcal{P}}(R_{\mathcal{R}}, S_{\mathcal{R}})$	$y \in I_{\rightarrow}(I_F(R_{\mathcal{R}}))(x) \Rightarrow y \in I_{\rightarrow}(I_F(S_{\mathcal{R}}))(x)$
$Sym(R)$	$Sym_{\mathcal{P}}(R_{\mathcal{R}})$	$y \in I_{\rightarrow}(I_F(R_{\mathcal{R}}))(x) \Rightarrow x \in I_{\rightarrow}(I_F(S_{\mathcal{R}}))(y)$
$Asy(R)$	$Asy_{\mathcal{P}}(R_{\mathcal{R}})$	$y \in I_{\rightarrow}(I_F(R_{\mathcal{R}}))(x) \Rightarrow x \notin I_{\rightarrow}(I_F(S_{\mathcal{R}}))(y)$
$Tra(R)$	$Tra_{\mathcal{P}}(R_{\mathcal{R}})$	$y \in I_{\rightarrow}(I_F(R_{\mathcal{R}}))(x) \wedge z \in I_{\rightarrow}(I_F(R_{\mathcal{R}}))(y) \Rightarrow$ $z \in I_{\rightarrow}(I_F(R_{\mathcal{R}}))(x)$
$Ref(R)$	$Ref_{\mathcal{P}}(R_{\mathcal{R}})$	$x \in I_{\rightarrow}(I_F(R_{\mathcal{R}}))(x)$
$Irr(R)$	$Irr_{\mathcal{P}}(R_{\mathcal{R}})$	$x \notin I_{\rightarrow}(I_F(R_{\mathcal{R}}))(x)$
$Dis(R,S)$	$Dis_{\mathcal{P}}(R_{\mathcal{R}}, S_{\mathcal{R}})$	$y \notin I_{\rightarrow}(I_F(R_{\mathcal{R}}))(x) \vee y \notin I_{\rightarrow}(I_F(S_{\mathcal{R}}))(x)$

Running Example To illustrate the mapping, we revisit the running example. A corresponding F-logic ontology looks as follows:

$$\begin{aligned}
\mathcal{T}^F =\{&Asset_{\mathcal{C}} :: Some(author_{\mathcal{R}}, \top_{\mathcal{C}}), Asset_{\mathcal{C}} :: AtMost(1, author_{\mathcal{R}}, \top_{\mathcal{C}}),\\
&Asset_{\mathcal{C}} :: AtMost(1, lastEditor_{\mathcal{R}}, \top_{\mathcal{C}}), Vocabulary :: Asset,\\
&subPropertyOf_{\mathcal{P}}(author_{\mathcal{R}}, editor_{\mathcal{R}}),\\
&subPropertyOf_{\mathcal{P}}(lastEditor_{\mathcal{R}}, editor_{\mathcal{R}})\}\\
\mathcal{A}^F =\{&Tom_{\mathcal{E}} : User_{\mathcal{C}}, Sarah_{\mathcal{E}} : User_{\mathcal{C}}, MetropolitanPlan_{\mathcal{E}} : Vocabulary_{\mathcal{C}}\}
\end{aligned}$$

Now we have to show that the mapping preservers entailment. First, we show that a formula θ is satisfiable in a $\mathcal{SROIQ}$ language $\mathcal{L}^{DL}$ if and only if a corresponding formula θ^F is satisfiable in a corresponding F-logic language $\mathcal{L}^F$.

Lemma 1. *Let θ be a formula in $\mathcal{L}^{DL}$ and θ^F a corresponding F-logic formula in an F-logic language $\mathcal{L}^F$. Then θ is satisfiable in some interpretation $\mathcal{I}$ of $\mathcal{L}^{DL}$ if and only if θ^F is satisfiable in some F-structure **I** of $\mathcal{L}^F$.*

Proof. The lemma can be proven by showing how an F-structure **I** can be constructed for a $\mathcal{SROIQ}$ interpretation $\mathcal{I}$ and vice versa. We will demonstrate that they will have the same truth value for the corresponding axioms.

Let us begin with the RBox axioms.

– $\mathcal{I}$ is a model of $Sym(R)$ iff $\langle x, y\rangle \in R^{\mathcal{I}}$ implies $\langle y, x\rangle \in R^{\mathcal{I}}$. We construct an F-structure **I** such that x, y are mapped to elements $I_F(x_{\mathcal{E}})$, $I_F(y_{\mathcal{E}})$ such

that $I_F(y_\mathcal{E}) \in I_\rightarrow(I_F(R_\mathcal{R}))(I_F(x_\mathcal{E}))$ implies $I_F(x_\mathcal{E}) \in I_\rightarrow(I_F(R_\mathcal{R}))(I_F(y_\mathcal{E}))$. Then, $\mathbf{I}$ is a model of $Sym_\mathcal{P}(R_\mathcal{R})$.
Conversely, suppose $\mathbf{J}$ is a model of $Sym_\mathcal{P}(R_\mathcal{R})$, thus, it has to adhere to the condition defined in Table 2. For such $I_F(x_\mathcal{E})$, $I_F(y_\mathcal{E})$, we can create an interpretation $\mathcal{J}$ with elements x, y such that $\langle x, y\rangle \in R^\mathcal{I}$ implies $\langle y, x\rangle \in R^\mathcal{I}$. Thus, the mapping for symmetric role is equivalent.

- The proof for Asy, Tra, Ref, Irr and Dis is analogous. One can easily verify the equivalence of conditions on an interpretation $\mathcal{I}$ described in [15] and the F-structure $\mathbf{I}$ conditions in Table 2.
- Consider a role inclusion axiom $R \sqsubseteq S$. For its model $\mathcal{I}$ must hold $R^\mathcal{I} \subseteq S^\mathcal{I}$. This can be expanded as $\langle x, y\rangle \in R^\mathcal{I}$ implies $\langle y, x\rangle \in S^\mathcal{I}$. Once again, we can construct an F-structure $\mathbf{I}$ containing elements $I_F(x_\mathcal{E})$ and $I_F(y_\mathcal{E})$, corresponding to x and y. Now, whenever $\langle x, y\rangle \in R^\mathcal{I}$, we will have $I_F(y_\mathcal{E}) \in I_\rightarrow(I_F(R_\mathcal{R}))(I_F(x_\mathcal{E}))$ in $\mathbf{I}$. The same for S and $S_\mathcal{R}$. This corresponds to the $\mathbf{I}$-condition for $subPropertyOf_\mathcal{P}(R_\mathcal{R}, S_\mathcal{R})$.
It should be clear that the converse holds as well. Thus, role inclusion axioms can be faithfully mapped to the $subPropertyOf_\mathcal{P}$ predicate and vice versa.

A general concept inclusion axiom (GCI) $C \sqsubseteq D$ is mapped to an is-a expression $C_\mathcal{C} :: D_\mathcal{C}$. For a DL model $\mathcal{I}$ must hold $C^\mathcal{I} \subseteq D^\mathcal{I}$. Written explicitly, this means that for all $x \in \Delta^\mathcal{I}$, $x \in C^\mathcal{I}$ implies $x \in D^\mathcal{I}$. In an F-structure $\mathbf{I}$ which is a model of $C_\mathcal{C} :: D_\mathcal{C}$ must hold $I_F(C_\mathcal{C}) \preceq I_F(D_\mathcal{C})$. The semantics defined in [18] specifies that if $x \in_U I_F(C_\mathcal{C})$ and $I_F(C_\mathcal{C}) \preceq I_F(D_\mathcal{C})$, then $x \in_U I_F(D_\mathcal{C})$. Simply put, in both DL and F-logic, the relationship is that whenever an element is an instance of a subconcept/subclass, it is also an instance of its superconcept/superclass. Therefore, the mapping between DL GCI axioms and subclass expressions in F-logic is equivalent. To ensure a correct baseline for GCI mapping, we will show the correspondence between the basic DL concept descriptions and their F-logic counterparts. Of particular interest are the newly introduced function symbols Not, $AtLeast$ etc. We have to ensure that the extensions of the corresponding concepts are equivalent. The extension of a DL concept C is the set of elements $C^\mathcal{I} \subseteq \Delta^\mathcal{I}$. In F-logic, a class' extension is the set of domain elements which are in $\in_U$ relation with its domain representation $I_F(C_\mathcal{C})$. Let us now process the concept mapping from Table 1.

- For C an atomic concept name the equivalence is trivial.
- For $C \leftarrow \neg D$, $(\neg D)^\mathcal{I} = \{x \in \Delta^\mathcal{I} \mid x \notin D^\mathcal{I}\}$. We can construct an F-structure $\mathbf{I}$ where we map $x^\mathcal{I}$ to $I_F(x_\mathcal{E})$, the fact that $x^\mathcal{I}$ does not belong into $D^\mathcal{I}$ would be mapped to $\mathbf{I}$ as the lack of $\in_U$-relationship between elements $I_F(x_\mathcal{E})$ and $I_F(D_\mathcal{C})$. Such elements represent the extension of $Not(D_\mathcal{C})$ which proves the DL to F-logic direction.
Now, for the other direction, the extension of $Not(D_\mathcal{C})$ in its model $\mathbf{J}$ is the set of all $I_F(x_\mathcal{E})$ such that $I_F(x_\mathcal{E}) \notin_U I_F(D_\mathcal{E})$. To build an equivalent $\mathcal{SROIQ}$ model $\mathcal{J}$, $I_F(x_\mathcal{E})$ will be mapped to $x^\mathcal{J}$, $I_F(D_\mathcal{E})$ to $D^\mathcal{J} \subset \Delta^\mathcal{J}$ such that $x^\mathcal{J} \notin D^\mathcal{J}$. These $x^\mathcal{J}$s represent the extension of $(\neg D)$, so the mapping allows to build equivalent models in both directions.

- For $C \leftarrow B \sqcap D$, its extension in a model $\mathcal{I}$ is the intersection of extensions of B and D, i.e., $(B \sqcap D)^{\mathcal{I}} = B^{\mathcal{I}} \cap D^{\mathcal{I}}$. In F-logic, the semantics of $(B_{\mathcal{C}}$ and $D_{\mathcal{C}})$ is the same, i.e., class intersection.
- For $C \leftarrow B \sqcup D$, the reasoning analogous to $B \sqcap D$.
- Take $C \leftarrow \geqslant nR.D$. For its instances $x \in \Delta^{\mathcal{I}}$ in a model $\mathcal{I}$ must hold $|\{y \in \Delta^{\mathcal{I}} \mid \langle x, y\rangle \in R^{\mathcal{I}} \wedge y \in D^{\mathcal{I}}\}| \geq n$. We can again construct an equivalent F-structure $\mathbf{I}$ where for each DL domain element y we create a new domain element y^F which belongs to $I_{\rightarrow}(I_F(R_{\mathcal{R}}))(I_F(x_{\mathcal{E}}))$ and is in a $\in_U$-relationship with $I_F(D_{\mathcal{C}})$. All such $x_{\mathcal{E}}$ constitute the extension of $AtLeast(n, R_{\mathcal{R}}, D_{\mathcal{C}})$.
- For $C \leftarrow \exists R.Self$, its extension in a model $\mathcal{I}$ is the set of all $x \in \Delta^{\mathcal{I}}$ such that $\langle x, x\rangle \in R^{\mathcal{I}}$. An equivalent F-structure $\mathbf{I}$ is built by mapping x to elements $I_F(x_{\mathcal{E}})$ such that $I_F(x_{\mathcal{E}}) \in I_{\rightarrow}(I_F(R_{\mathcal{R}}))(I_F(x_{\mathcal{E}}))$. Such elements comprise the extension of $HasSelf(R_{\mathcal{R}})$
- For $C \leftarrow \{a\}$, $(\{a\})^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid x = a^{\mathcal{I}}\}$ in a model $\mathcal{I}$. The interpretation of $Nom(a_{\mathcal{E}})$ also relies on equality of domain elements.
- Take $C \leftarrow \forall R.D$ and a model $\mathcal{I}$, $(\forall R.D)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \forall y \in \Delta^{\mathcal{I}}$ s.t. $\langle x, y\rangle \in R^{\mathcal{I}} \Rightarrow y \in D^{\mathcal{I}}\}$. x and y are mapped to F-structure $\mathbf{I}$ elements $I_F(x_{\mathcal{E}})$ and $I_F(y_{\mathcal{E}})$ in such a way that $\forall I_F(y_{\mathcal{E}}) \in I_{\rightarrow}(I_F(R_{\mathcal{R}}))(I_F(x_{\mathcal{E}}))$ holds that $I_F(y_{\mathcal{E}}) \in_U I_F(D_{\mathcal{C}})$. Such an F-structure provides an extension for $All(R_{\mathcal{R}}, D_{\mathcal{C}})$. Clearly, this transformation also works in the other direction.
- Finally, let $C \leftarrow \exists R.D$. Its extension in a model $\mathcal{I}$ corresponds to $(\exists R.D)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \exists\, y \in \Delta^{\mathcal{I}}$ s.t. $\langle x, y\rangle \in R^{\mathcal{I}} \wedge y \in D^{\mathcal{I}}\}$. A corresponding F-structure $\mathbf{I}$ contains an element $I_F(y_{\mathcal{E}})$ such that $I_F(y_{\mathcal{E}}) \in I_{\rightarrow}(I_F(R_{\mathcal{R}}))(I_F(x_{\mathcal{E}})) \wedge I_F(y_{\mathcal{E}}) \in_U I_F(D_{\mathcal{C}})$. It can be seen that this transformation again works in both directions. Moreover $\mathbf{I}$ provides an extension of $Some(R_{\mathcal{R}}, D_{\mathcal{C}})$, so we can conclude that the mapping is again faithful.

The last part of this proof needs to deal with ABox axioms. However, ABox axioms can be *internalized* using nominals as follows: $C(a)$ to $\{a\} \sqsubseteq C$, $R(a, b)$ to $\{a\} \sqsubseteq \exists R.\{b\}$, $a = b$ to $\{a\} \equiv \{b\}$[4] and $a \neq b$ to $\{a\} \not\equiv \{b\}$. This way, there is no need to treat them explicitly. □

The lemma allows us to show that entailment is preserved by the mapping.

Theorem 1. *Let Θ and Θ^F be corresponding theories in $\mathcal{L}^{DL}$ and $\mathcal{L}^F$. For any formula θ in $\mathcal{L}^{DL}$ holds:*

$$\Theta \models \theta \textit{ iff } \Theta^F \models^F \theta^F,$$

where $\models^F$ represents F-logic entailment.

Proof. The proof relies on Lemma 1 and the fact that entailment checking can be reduced to satisfiability checking. □

[4] $C \equiv D$ corresponds to $C \sqsubseteq D \wedge D \sqsubseteq C$.

4 Mapping Integrity Constraints

Integrity constraints mapping between $\mathcal{SROIQ}$ and F-logic consists of two parts: 1) IC semantics with closed-world view of the data; 2) means of validating these ICs. The mapping is important because the application object model is based on the integrity constraints.

4.1 Integrity Constraints Semantics

IC semantics allows to impose a closed-world view on a portion of the knowledge base $\mathcal{K}$ affected by the integrity constraints. We follow the approach of Tao et al. [28] and define an augmented F-structure $\mathbf{I}^{IC}$ with IC semantics. This approach has the advantage of not introducing additional syntactic constructs and giving the IC axioms a natural, easy-to-understand meaning. The semantics uses the notion of *minimal equality models* (Mod_{ME}) to support a *weak* form of unique name assumption. The original definition of Mod_{ME} from [28] can be carried over to F-logic as follows:

Consider a knowledge base $\mathcal{K}^F$ and let $E_{\mathbf{I}}$ be the set of equality relations satisfied by $\mathbf{I}$, i.e., $E_{\mathbf{I}} = \{\langle a, b\rangle \mid a, b \in \mathcal{E} \text{ s.t. } \mathbf{I} \models^F I_F(a) = I_F(b)\}$. A relation $\mathbf{I} \prec^F_= \mathbf{J}$, where $\mathbf{I}$ and $\mathbf{J}$ are F-structures, holds iff:

- $\forall C \in \mathcal{C} \cup \mathcal{A}$, if $\mathbf{I} \models^F a\!:\!C$, then $\mathbf{J} \models^F a\!:\!C$,
- $\forall R \in \mathcal{R}$, if $\mathbf{I} \models^F a[R \rightarrow b]$, then $\mathbf{J} \models^F a[R \rightarrow b]$,
- $E_{\mathbf{I}} \subset E_{\mathbf{J}}$.

$Mod^F_{ME}(\mathcal{K}^F)$ is then defined as $Mod^F_{ME}(\mathcal{K}^F) = \{\mathbf{I} \mid \mathbf{I} \text{ is a model of } \mathcal{K}^F \text{ s.t. } \nexists \mathbf{J}\ E_{\mathbf{J}} \prec^F_= E_{\mathbf{I}}\}$.

The augmented F-structure with IC semantics is a tuple $\mathbf{I}^{IC} = \langle U, \prec^{IC}_U, \in^{IC}_U, I_F, I^{IC}_{\mathcal{P}}, I^{IC}_{\rightarrow}, I_{\Rightarrow}\rangle$, where:

- $\prec^{IC}_U = \{\langle I_F(x), I_F)(y)\rangle \mid x, y \in \mathcal{C} \text{ s.t. } \forall \mathcal{J} \in Mod^F_{ME}(\mathcal{K}^F), \mathcal{J} \models^F I_F(x) \prec_U I_F(y)\}$
- $\in^{IC}_U = \{\langle I_F(x), I_F(y)\rangle \mid x \in \mathcal{E}, y \in \mathcal{C} \cup \mathcal{A} \text{ s.t. } \forall \mathcal{J} \in Mod^F_{ME}(\mathcal{K}^F), \mathcal{J} \models^F I_F(x) \in_U I_F(y)\}$
- $I_F(y) \in I^{IC}_{\rightarrow}(I_F(z))(I_F(x))$ iff $x, y \in \mathcal{E}, z \in \mathcal{R} \wedge \forall \mathcal{J} \in Mod^F_{ME}(\mathcal{K}^F), \mathcal{J} \models^F I_F(y) \in I^{IC}_{\rightarrow}(I_F(z))(I_F(x))$
- $I^{IC}_{\mathcal{P}}(p) = \{\langle I_F(y_1), ..., I_F(y_n)\rangle \mid y_i \in \mathcal{F} \text{ s.t. } \forall \mathcal{J} \in Mod^F_{ME}(\mathcal{K}^F), \mathcal{J} \models^F \langle I_F(y_1), ..., I_F(y_n)\rangle \in I^{IC}_{\mathcal{P}}(p)\}$, where n is the arity of p,
- And the other parts of $\mathbf{I}^{IC}$ are the same as in a regular F-structure.

Based on $\mathbf{I}^{IC}$, we can now define the IC semantics of concept descriptions. This is done in Table 3. One modification is the switch from $All(R_{\mathcal{R}}, C_{\mathcal{C}})$ to the built-in signature expression $[R_{\mathcal{R}} \Rightarrow C_{\mathcal{C}}]$. This can be done thanks to the notion of *typing*, which requires data expressions to correspond to a signature expression declaring their types, e.g., for signature $C_{\mathcal{C}}[R_{\mathcal{R}} \Rightarrow D_{\mathcal{C}}]$, typing rules require d from data expression $c[R_{\mathcal{R}} \rightarrow d]$ to be of type $D_{\mathcal{C}}$, i.e., $d\!:\!D_{\mathcal{C}}$. Typing is an optional, non-monotonic, part of F-logic. We utilize it for IC declaration because it provides a nice, succinct, frame-based syntax. IC Semantics of axioms should follow trivially from the definitions in Table 3.

Table 3. Integrity constraint semantics of F-logic concept descriptions. The right hand column specifies a condition under which an individual x is an instance of the concept specified in the left hand column under the IC semantics.

Concept	$\mathbf{I}^{IC} \models^F x : Concept$ iff
$Not(C_\mathcal{C})$	$x \in \mathcal{E} \wedge I_F(x) \notin_U^{IC} I_F(C_\mathcal{C})$
$C_\mathcal{C}$ and $D_\mathcal{C}$	$x \in \mathcal{E} \wedge I_F(x) \in_U^{IC} I_F(C_\mathcal{C}) \wedge I_F(x) \in_U^{IC} I_F(D_\mathcal{C})$
$C_\mathcal{C}$ or $D_\mathcal{C}$	$x \in \mathcal{E} \wedge I_F(x) \in_U^{IC} I_F(C_\mathcal{C}) \vee I_F(x) \in_U^{IC} I_F(D_\mathcal{C})$
$AtLeast(n, R_\mathcal{R}, C_\mathcal{C})$	$x \in \mathcal{E} \wedge \exists y_1, ...y_n \in \mathcal{E}$ s.t. $I_F(y_i) \in I_\rightarrow^{IC}(I_F(R_\mathcal{R}))(I_F(x))$
	$\wedge I_F(y_i) \in_U^{IC} I_F(C_\mathcal{C}) \wedge \neg(I_F(y_i) = I_F(y_j))$
$HasSelf(R_\mathcal{R})$	$x \in \mathcal{E} \wedge I_F(x) \in I_\rightarrow^{IC}(I_F(R_\mathcal{R}))(I_F(x))$
$Nom(a_\mathcal{E})$	$x \in \mathcal{E} \wedge I_F(x) = I_F(a_\mathcal{E})$
$[R_\mathcal{R} \Rightarrow D_\mathcal{C}]$	$\mathbf{I}^{IC}$ is a *typed* F-structure [18] (Sec. 13)
$Some(R_\mathcal{R}, C_\mathcal{C})$	$x \in \mathcal{E} \wedge \exists y \in \mathcal{E}$ s.t. $I_F(y) \in I_\rightarrow^{IC}(I_F(R_\mathcal{R}))(I_F(x)) \wedge I_F(y) \in_U^{IC} I_F(C_\mathcal{C})$

Running Example Reviewing our running example, the biggest change is the use of method signatures with cardinality constraints. This significantly reduces the verbosity of the ICs and makes them arguable easier to understand.

$$\mathcal{IC}^F = \{Vocabulary_\mathcal{C}[author_\mathcal{R} \Rightarrow_{\{1:1\}} User_\mathcal{C}; lastEditor_\mathcal{C} \Rightarrow_{\{0:1\}} User_\mathcal{C}]\}$$

Figure 1 illustrates how a resulting integrity constraints-based model may look in terms of a UML class diagram.

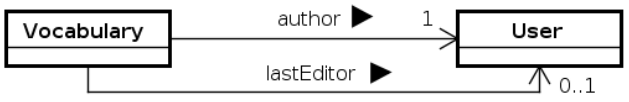

Fig. 1. UML class diagram of a model based on the integrity constraints from the running example.

4.2 Integrity Constraints Validation

Now that one is able to define integrity constraints for an ontology, it is necessary to be able to validate them as well. IC semantics is a convenient construct, but because no corresponding implementation exists, it is impractical. Thus, integrity constraint validation in F-logic utilizes the built-in possibility to execute queries over the underlying ontology. F-logic is a full-fledged logic programming language, so it allows to define rules and pose queries to the knowledge base.

Since ICs represent a close-world view of the ontology, *negation as failure* (NAF) is necessary to be able to represent it in the queries. Like most logic programming languages [23], we introduce the NAF operator **not**, whose semantics is $\mathcal{K}^F \models \mathbf{not}(\alpha)$ iff $\mathcal{K}^F \not\models \alpha$, where α is an F-formula.

We now show how the IC axioms can be translated into F-logic queries with **not**. The rationale is that if the knowledge base entails the query, there is an IC violation. We again follow the line of reasoning of [28], which introduces two translation operators: $\mathcal{T}_C$ for concepts and $\mathcal{T}$ for axioms. Definition of $\mathcal{T}_C$ can be found in Table 4, $\mathcal{T}$ is then presented in Table 5.

Table 4. Integrity constraints validation transformation rules for concepts. C_A is an atomic class name.

Concept	$\mathcal{T}_C$
$\mathcal{T}_C(x\!:\!C_A)$	$x\!:\!C_A$
$\mathcal{T}_C(x\!:\!Not(C))$	$\mathbf{not}(x\!:\!\mathcal{T}_C(C))$
$\mathcal{T}_C(x\!:\!(C_1 \text{ and } C_2))$	$x\!:\!\mathcal{T}_C(C_1) \wedge x\!:\!\mathcal{T}_C(C_2)$
$\mathcal{T}_C(x\!:\!(C_1 \text{ or } C_2))$	$x\!:\!\mathcal{T}_C(C_1) \vee x\!:\!\mathcal{T}_C(C_2)$
$\mathcal{T}_C(x\!:\!AtLeast(n,R,C))$	$\bigwedge_{1\le i\le n} x[R \rightarrow y_i] \wedge y_i\!:\!\mathcal{T}_C(C) \bigwedge_{1\le i\le j\le n} \mathbf{not}(y_i = y_j)$
$\mathcal{T}_C(x\!:\!HasSelf(R))$	$x[R \rightarrow x]$
$\mathcal{T}_C(x\!:\!Nom(a))$	$x = a$
$\mathcal{T}_C(x[R \Rightarrow C])$	$x[R \rightarrow y] \Rightarrow y\!:\!\mathcal{T}_C(C)$
$\mathcal{T}_C(x\!:\!Some(R,C))$	$x[R \rightarrow y] \wedge y\!:\!\mathcal{T}_C(C)$

The universal role restriction concept comes with a little twist. Instead of representing a standalone concept, we attach a corresponding signature expression to the target concept, i.e., instead of mapping a GCI $C \sqsubseteq \forall R.D$, we have directly $C_{\mathcal{C}}[R_{\mathcal{R}} \Rightarrow D_{\mathcal{C}}]$. A corresponding IC validation query in F-logic is created by verifying that data expressions of all instances of $C_{\mathcal{C}}$ comply with the signature expression, i.e., $\mathcal{T}_C(x[R_{\mathcal{R}} \Rightarrow D_{\mathcal{C}}])$. This can be also seen in the running example below.

Table 5. Integrity constraints validation transformation rules for axioms. C_i is a concept, R_i is a role and x, y_i are variables.

Axiom	$\mathcal{T}$
$\mathcal{T}(C_1 :: C_2)$	$\mathcal{T}_C(x\!:\!C_1) \wedge \mathbf{not}(\mathcal{T}_C(x\!:\!C_2))$
$\mathcal{T}(subPropertyOf_{\mathcal{P}}(R_1,R_2))$	$x[R_1 \rightarrow y] \wedge \mathbf{not}(x[R_2 \rightarrow y])$
$\mathcal{T}(Sym_{\mathcal{P}}(R))$	$x[R \rightarrow y] \wedge \mathbf{not}(y[R \rightarrow x])$
$\mathcal{T}(Asy_{\mathcal{P}}(R))$	$x[R \rightarrow y] \wedge y[R \rightarrow x]$
$\mathcal{T}(Tra_{\mathcal{P}}(R))$	$x[R \rightarrow y] \wedge y[R \rightarrow z] \wedge \mathbf{not}(x[R \rightarrow z])$
$\mathcal{T}(Ref_{\mathcal{P}}(R))$	$\mathbf{not}(x[R \rightarrow x])$
$\mathcal{T}(Irr_{\mathcal{P}}(R))$	$x[R \rightarrow x]$
$\mathcal{T}(Dis_{\mathcal{P}}(R_1,R_2))$	$x[R_1 \rightarrow y] \wedge x[R_2 \rightarrow y]$

Running Example Since a signature expression with cardinality constraints is essentially a combination of multiple concept descriptions, it results in multiple validation queries. The queries presented below can be executed in F-logic implementations like FLORA-2[5], which already supports the NAF operator. Presence of results for any of the queries indicates an IC violation. In our case, the constraints are violated by the lack of an explicit author of $MetropolitanPlan$, which is manifested by the second query below. Asserting an author, e.g., $MetropolitanPlan_{\mathcal{E}}[author_{\mathcal{R}} \rightarrow Tom_{\mathcal{E}}]$, would fix the IC violation.

$$\begin{aligned}\mathcal{T} =\{ & x:Vocabulary \wedge x[author_{\mathcal{R}} \rightarrow y] \wedge \mathbf{not}(y:User_{\mathcal{C}}),\\ & x:Vocabulary_{\mathcal{C}} \wedge \mathbf{not}(x[author_{\mathcal{R}} \rightarrow y] \wedge y:User_{\mathcal{C}}),\\ & x:Vocabulary_{\mathcal{C}} \wedge x[author_{\mathcal{R}} \rightarrow \{y_1, y_2\}] \bigwedge_{1 \leq i \leq 2} y_i:User_{\mathcal{C}} \wedge \mathbf{not}(y_1 = y_2)\\ & x:Vocabulary \wedge x[lastEditor_{\mathcal{R}} \rightarrow y] \wedge \mathbf{not}(y:User_{\mathcal{C}}),\\ & x:Vocabulary_{\mathcal{C}} \wedge x[lastEditor_{\mathcal{R}} \rightarrow \{y_1, y_2\}] \bigwedge_{1 \leq i \leq 2} y_i:User_{\mathcal{C}} \wedge \mathbf{not}(y_1 = y_2)\}\end{aligned}$$

Finally, we have to show that the validation queries faithfully represent the IC semantics, i.e., that validation queries generated from IC axioms return results whenever any of the IC axioms are violated by the knowledge base.

Theorem 2. *Consider a knowledge base $\mathcal{K}$, a set of integrity constraint axioms $\mathcal{IC}$ and a set of IC validation queries $\mathcal{Q}$, constructed by applying the translation operator $\mathcal{T}$ on each IC axiom α in $\mathcal{IC}$. If $\mathcal{K}$ violates any of the IC axioms in $\mathcal{IC}$, then $\exists\ q \in \mathcal{Q}$ such that $\mathcal{K} \models^F q$.*

Proof. We first show the equivalence of RBox IC axioms to their validation queries.

- The IC semantics of axiom $\alpha = subPropertyOf_{\mathcal{P}}(R, Q)$ requires an F-structure $\mathbf{I}^{IC}$ to satisfy $I_F(y) \in I^{IC}_{\rightarrow}(I_F(R))(I_F(x)) \Rightarrow I_F(y) \in I^{IC}_{\rightarrow}(I_F(Q))(I_F(x))$ for all individuals $x, y \in \mathcal{E}$. Recall that $\mathbf{I}^{IC}$ is the smallest model w.r.t. equality. For the axiom to be violated, we need to find a model in which the axiom is not true, i.e., an F-structure $\mathbf{J}$ such that $\mathbf{J} \models^F \exists x, y \in \mathcal{E}$ s.t. $I_F(y) \in I_{\rightarrow}(I_F(R))(I_F(x)) \wedge I_F(y) \notin I_{\rightarrow}(I_F(Q))(I_F(x))$.
  Now, the corresponding validation query, as specified in Table 5, is $x[R \rightarrow y] \wedge \mathbf{not}(x[Q \rightarrow y])$, where **not** represents the absence of knowledge. The query is looking for an F-structure $\mathbf{H} \models^F I_F(y) \in I_{\rightarrow}(I_F(R))(I_F(x))$ and $\mathbf{H} \not\models^F I_F(y) \in I_{\rightarrow}(I_F(Q))(I_F(x))$, where $x, y \in \mathcal{E}$ are existentially quantified. Thus, the two parts can be combined as follows: $\mathbf{H} \models^F \exists x, y \in \mathcal{E}$ s.t. $I_F(y) \in I_{\rightarrow}(I_F(R))(I_F(x)) \wedge I_F(y) \notin I_{\rightarrow}(I_F(Q))(I_F(x))$. It can be seen that the query returns results if and only if integrity constraint axiom α is violated.
- For $\alpha = Sym_{\mathcal{P}}(R)$, IC semantics demands an F-structure $\mathbf{I}^{IC} \models^F I_F(y) \in I^{IC}_{\rightarrow}(I_F(R))(I_F(x)) \Rightarrow I_F(x) \in I^{IC}_{\rightarrow}(I_F(R))(I_F(y))$. A series of transformations analogous to the sub-property case above will lead to IC violation in case of an F-structure $\mathbf{J}$ such that $\mathbf{J} \models^F \exists x, y \in \mathcal{E}$ s.t. $I_F(y) \in$

[5] `http://flora.sourceforge.net/`, accessed 2019-04-10.

$I_{\rightarrow}(I_F(R))(I_F(x)) \wedge I_F(x) \notin I_{\rightarrow}(I_F(R))(I_F(y))$ and is a model of the knowledge base containing IC axiom α.
The validation query $x[R \rightarrow y] \wedge \mathbf{not}(y[R \rightarrow x])$ will be looking for an F-structure $\mathbf{H}$ such that $\mathbf{H} \models^F \exists x, y \in \mathcal{E}$ s.t. $I_F(y) \in I_{\rightarrow}(I_F(R))(I_F(x)) \wedge I_F(x) \notin I_{\rightarrow}(I_F(R))(I_F(y))$. Thus, we have again arrived at equivalent formulas for IC violation.
– IC axiom and validation query equivalence for $\alpha = Asy_{\mathcal{P}}(R)$ ($\alpha = Tra_{\mathcal{P}}(R)$, $\alpha = Ref_{\mathcal{P}}(R)$, $\alpha = Irr_{\mathcal{P}}(R)$, and $\alpha = Dis_{\mathcal{P}}(R, Q)$) are proven analogously.

The second part of the proof deals with general concept inclusion axioms for various concept descriptions. Once again, the goal is to show that, for each concept description, the GCI axiom, when taken as an integrity constraint, is violated if and only if a corresponding validation query, constructed using the transformation rules from Table 4, finds results.

– Let $\alpha = C :: Not(D)$ be an integrity constraint axiom. An F-structure $\mathbf{I}^{IC}$ satisfies it if $\mathbf{I}^{IC} \models^F I_F(C) \preceq_U^{IC} I_F(Not(D))$. α is violated if there exists an F-structure $\mathbf{J}$ such that $\mathbf{J} \not\models^F I_F(C) \preceq_U I_F(Not(D))$, i.e., $\mathbf{J} \models^F \neg(I_F(C) \preceq_U I_F(Not(D)))$. Now, from the properties of F-structures, we know that if $a \in_U b \wedge b \preceq_U c$, then $a \in_U c$ [18]. So, in $\mathbf{J}$, we have that $\neg(I_F(C) \preceq_U I_F(Not(D)))$ iff $\exists x \in \mathcal{E}$ s.t. $I_F(x) \in_U I_F(C) \wedge x \notin_U I_F(Not(D))$. Given the interpretation of $Not(D)$, this can be rewritten as $\exists x \in \mathcal{E}$ s.t. $I_F(x) \in_U I_F(C) \wedge x \in_U I_F(D)$.
The validation query for GCI involving $Not(D)$ is as follows: $x : C \wedge \mathbf{not}(x : Not(D))$. Thus, it searches for an F-structure $\mathbf{H}$ such that $\mathbf{H} \models^F I_F(x) \in_U I_F(C)$ and $\mathbf{H} \not\models^F I_F(x) \in_U I_F(Not(D))$, with $x \in \mathcal{E}$ existentially quantified. This can be rewritten as $\mathbf{H} \models^F \exists x \in \mathcal{E}$ s.t. $I_F(x) \in_U I_F(C) \wedge I_F(x) \notin_U I_F(Not(D))$. Again, flipping the $\notin_U$ and $Not(D)$, the result is $\mathbf{H} \models^F \exists x \in \mathcal{E}$ s.t. $I_F(x) \in_U I_F(C) \wedge I_F(x) \in_U I_F(D)$. So the query has results iff α is violated.
– For an IC axiom $\alpha = C :: (B \text{ and } D)$ an F-structure $\mathbf{I}^{IC}$ must satisfy $I_F(C) \preceq_U^{IC} I_F(B) \wedge I_F(C) \preceq_U^{IC} I_F(D)$. α is violated if there exists a model of the knowledge base $\mathbf{J}$ such that $\mathbf{J} \models^F \exists x \in \mathcal{E}$ s.t. $I_F(x) \in_U I_F(C) \wedge \neg(I_F(x) \in_U I_F(B) \wedge I_F(x) \in_U I_F(D))$, so, $\mathbf{J} \models^F \exists x \in \mathcal{E}$ s.t. $I_F(x) \in_U I_F(C) \wedge (I_F(x) \notin_U I_F(B) \vee I_F(x) \notin_U I_F(D))$.
The validation query for α – $x : C \wedge \mathbf{not}(x : (B \text{ and } D))$ – searches for an F-structure $\mathbf{H}$ such that $\mathbf{H} \models^F I_F(x) \in_U I_F(C)$ and $\mathbf{H} \not\models^F I_F(x) \in_U I_F(B) \wedge I_F(x) \in_U I_F(D)$, where x is existentially quantified over both formulas. This is again combined into a single query formula $\mathbf{H} \models^F \exists x \in \mathcal{E}$ s.t. $I_F(x) \in_U C \wedge (I_F(x) \notin_U I_F(B) \vee I_F(x) \notin_U I_F(D))$. Once again, the query corresponds to the formula for violation of α.
– For an IC axiom $\alpha = C :: (B \text{ or } D)$, the proof goes along the same lines as for $C :: (B \text{ and } D)$. Thus, α is violated if there is a model $\mathbf{J}$ such that $\mathbf{J} \models^F \exists x \in \mathcal{E}$ s.t. $I_F(x) \in_U I_F(C) \wedge I_F(x) \notin_U I_F(B) \wedge I_F(x) \notin_U I_F(D)$.
Similarly, the validation query $x : C \wedge \mathbf{not}(x : (B \text{ or } D))$ searches for an F-structure $\mathbf{H}$ such that $\mathbf{H} \models^F \exists x \in \mathcal{E}$ s.t. $I_F(x) \in_U I_F(C) \wedge I_F(x) \notin_U$

$I_F(B) \wedge I_F(x) \notin_U I_F(D)$, so the validation query corresponds to an IC violation.

– Take an IC axiom $\alpha = C :: \mathit{AtLeast}(n, R, D)$. To be satisfied, it requires an F-structure $\mathbf{I}^{IC}$ such that $\mathbf{I}^{IC} \models^F I_F(x) \in_U^{IC} I_F(C) \rightarrow (\exists y_{1..k} \in \mathcal{E}$ s.t. $I_F(y_i) \in I_{\rightarrow}(I_F(R))(I_F(x)) \wedge I_F(y_i) \in_U^{IC} I_F(D) \wedge \neg(I_F(y_i) = I_F(y_j))$ for $k \geq n$. Now, for α to be violated, we have to find a model $\mathbf{J}$ in which $k < n$. Thus, α is violated if $\mathbf{J} \models^F \exists x \in \mathcal{E}$ s.t. $I_F(x) \in_U^{IC} I_F(C) \wedge \exists y_{1..k} \in \mathcal{E}$ s.t. $I_F(y_i) \in I_{\rightarrow}(I_F(R))(I_F(x)) \wedge I_F(y_i) \in_U^{IC} I_F(D) \wedge \neg(I_F(y_i) = I_F(y_j)$ for $k < n$. Here, k is treated as maximum, i.e., there can be no more that k ys. α is validated by $x : C \wedge \mathbf{not}(\bigwedge_{1 \leq i \leq n} x[R \rightarrow y_i] \wedge y_i : D \bigwedge_{1 \leq i \leq j \leq n} \mathbf{not}(y_i = y_j))$. Let us now rewrite it at the F-structure level. We will also replace the outer big conjunction with existential quantifier (makes no difference in terms of semantics) to allow us to match it to the IC violation formula. Thus, the query is looking for a model $\mathbf{H}$ such that $\mathbf{H} \models^F \exists x \in \mathcal{E}$ s.t. $I_F(x) \in_U I_F(C) \wedge \exists y_{1..l} \in \mathcal{E}$ s.t. $I_F(y_i) \in I_{\rightarrow}(I_F(R))(I_F(x)) \wedge I_F(y) \in_U I_F(D) \wedge \bigwedge_{1 \leq i \leq j \leq n} \mathbf{not}(I_F(y_i) = I_F(y_j))$ for $l < n$. Again, l is a maximum, so no more unique ys can exist.

– Let $\alpha = C :: \mathit{HasSelf}(R)$ be an IC axiom. It is violated if we find a model $\mathbf{J}$ such that $\mathbf{J} \models^F \exists x \in \mathcal{E}$ s.t. $I_F(x) \in I_F(C) \wedge I_F(x) \notin I_{\rightarrow}(I_F(R))(I_F(x))$.
The corresponding validation query searches for a model $\mathbf{H}$ for which it holds that $\mathbf{H} \models^F I_F(x) \in_U I_F(C)$ and $\mathbf{H} \not\models^F I_F(x) \in_U I_F(\mathit{HasSelf}(R))$ for some x from $\mathcal{E}$. Putting the formulas together and replacing $\mathit{HasSelf}(R)$ with the corresponding interpretation, we get $\mathbf{H} \models^F \exists x \in \mathcal{E}$ s.t. $I_F(x) \in_U I_F(C) \wedge I_F(x) \notin I_{\rightarrow}(I_F(R))(I_F(x))$.

– An IC axiom $\alpha = C :: \mathit{Nom}(a)$ is violated in a model $\mathbf{J}$ such that $\mathbf{J} \models^F \exists x \in \mathcal{E}$ s.t. $I_F(x) \in I_F(C) \wedge I_F(x) \neq I_F(a)$. It should be easy to see that the same formula can be constructed from the corresponding validation query.

– As was stated earlier, universal quantification integrity constraints are written using signature expressions in F-logic. Thus, an IC axiom α now takes the form of $\alpha = C[R \Rightarrow D]$. This requires an IC model $\mathbf{I}^{IC}$ to satisfy $I_F(x) \in_U^{IC} I_F(C) \Rightarrow (I_F(y) \in I_{\rightarrow}^{IC}(I_F(R))(I_F(x)) \Rightarrow I_F(y) \in_U^{IC} I_F(D))$ for any $x, y \in \mathcal{E}$. α is violated in a model $\mathbf{J}$ if $\mathbf{J} \models^F \exists x, y \in \mathcal{E}$ s.t. $I_F(x) \in_U I_F(C) \wedge I_F(y) \in I_{\rightarrow}(I_F(R))(I_F(x)) \wedge I_F(y) \notin_U I_F(D)$.
Now, for the validation query, we have $\mathcal{T}(C[R \Rightarrow D]) = x : C \wedge \mathbf{not}(x[R \Rightarrow D])$ (recall that signature expressions propagate from classes to instances). According to well-typing rules [18], a matching F-structure $\mathbf{H}$ satisfies $I_F(x) \in_U I_F(C) \wedge \neg(I_F(y) \in_U I_{\rightarrow}(I_F(R))(I_F(x)) \Rightarrow I_F(y) \in_U I_F(D))$, thus $I_F(x) \in_U I_F(C) \wedge I_F(y) \in_U I_{\rightarrow}(I_F(R))(I_F(x)) \wedge I_F(y) \notin_U I_F(D)$ for some $x, y \in \mathcal{E}$.

– Lastly, consider an integrity constraint axiom $\alpha = C :: \mathit{Some}(R, D)$, which requires F-structures to satisfy $\forall x \in \mathcal{E}\ I_F(x) \in_U^{IC} I_F(C) \Rightarrow \exists y \in \mathcal{E}$ s.t. $I_F(y) \in I_{\rightarrow}^{IC}(I_F(R))(I_F(x)) \wedge I_F(y) \in_U^{IC} I_F(D)$. α is violated if we find a model $\mathbf{J}$ such that $\mathbf{J} \models^F \exists x \in \mathcal{E}$ s.t. $I_F(x) \in_U I_F(C) \wedge \forall y \in \mathcal{E}\ I_F(y) \notin I_{\rightarrow}(I_F(R))(I_F(x)) \vee I_F(y) \notin_U I_F(D)$.
Transformation of validation query $\mathcal{T}(C :: \mathit{Some}(R, D)) = x : C \wedge \mathbf{not}(x : \mathit{Some}(R, D))$ follows the path treated several times above and arrives at the

same expression as the IC α violation, i.e. $\exists x \in \mathcal{E}$ s.t. $I_F(x) \in_U I_F(C) \wedge \forall y \in \mathcal{E}\ I_F(y) \notin I_{\rightarrow}(I_F(R))(I_F(x)) \vee I_F(y) \notin_U I_F(D)$.

Thus, we have shown that for both RBox axioms and the GCI axiom involving various concept descriptions integrity constraint checking can be reduced to query answering in F-logic. □

5 Related Work

This section reviews works concerning application access to DL ontologies, provides a comparison of approaches to mapping between description logics and F-logic and discusses methods of closed-world reasoning.

5.1 Application Access to Ontologies

There exists a number of application libraries which provide programmatic access to ontologies. They can be roughly divided into two groups [21]:

Domain-independent APIs Jena [8], OWL API [14], or RDF4J [5] are low-level libraries which represent ontological data on the axiom or statement level.

Domain-specific APIs Empire [13], KOMMA [30], ActiveRDF [25], and RDFReactor [29] allow the application to access ontological data in a frame-based manner.[6]

Libraries of type 1 are suitable for generic applications like ontology editors or vocabulary explorers, but their use in domain-specific applications is cumbersome, because they require a lot of boilerplate code to allow dealing with higher-level business objects. Libraries of type 2 employ some kind of object-ontological mapping (sometimes also called object-triple mapping), so that they map ontological concepts to programming language reference types, properties to attributes etc. The problem with these libraries is that they often do not take into account the open-world nature of ontologies. They do not deal with inferred knowledge (an inferred assertion cannot be directly removed), and the mapping is done without any formal basis. These libraries rarely support knowledge outside the mapped object model and tend to have issues with individual identity. For instance, given an OWL ontology $\mathcal{O} = \{Vocabulary \sqsubseteq Asset, Vocabulary(a)\}$, Empire, when asked to retrieve a twice - as an $Asset$ and as a $Vocabulary$, will return two different objects. The consequences can be anywhere between overwriting updates to deletion of an object that is being used.

[6] A detailed comparison of these libraries can be found in [22].

5.2 Mapping between DLs and F-logic

The relationship between description logics and F-logic can be approached from different directions. One, which has been investigated in [12] or [16], enriches a DL knowledge base with (F-logic) rules to provide additional or more efficient inferences ([12] considers logic programming languages in general).

The other direction tries to develop a mapping between the two languages. [7] exploits the fact that DLs are a subset of FOL and maps them to the FOL flavor of F-logic, i.e., concepts to unary predicates and roles to binary predicates. On the other hand, Balaban [3] attempts to map DL constructs to F-logic frames. However, his article deals only with less expressive DLs ($\mathcal{ALC}$). Close to our approach is also the work of de Bruijn and Heymans [6] which maps $\mathcal{SHIQ}$ to F-logic by first translating it to predicate-based FOL and then mapping it to F-logic. Compared to Balaban, our mapping deals with more expressive languages and considers the mapping of integrity constraints. de Bruijn and Heymans' work presents, in our opinion, a less readable, although arguably more straightforward, approach to the mapping. The authors of F-logic themselves discuss its potential as ontology-modeling language in [17,31].

5.3 Closed-world Reasoning

Application of integrity constraints to DL ontologies, as discussed in Section 2.2, is closely related to *(local) closed-world reasoning*. Significant amount of work has been done in this area in connection with rule-based languages. They often split the knowledge base into a DL-based OWA part and a rule-based CWA part with stable [11,9] or well-founded [19,9] model semantics.

Another approach similar to [26] is based on *grounded circumscription* where selected concepts and roles are closed and minimized, i.e., they contain only the minimum necessary *known* individuals [27].

6 Conclusions

We have introduced a novel formalism for object-ontological mapping based on the description logic $\mathcal{SROIQ}$ and F-logic. The formalism maps both a DL ontology and integrity constraints which provide a closed-world view of (a portion of) the ontology. We have shown that the mapping preserves entailment and presented means of validating the integrity constraints. As has been shown in [20], integrity constraints represent the basis of the contract between an ontology and an object model and are used to define the object model.

However, the presented work is just the first step. The mapping represents a static structure of the model and the data. The next step should be defining operations over the data in terms of the formalism. With such definitions, ontological operations like data retrieval or modifications would have predictable and well defined results.

Another step is the actual translation of the F-logic intermediate model into an object model in a mainstream object-oriented programming language like Java. Finally, the operations need to be implemented according to the definitions.

Acknowledgment This work was supported by grant No. SGS19/110/OHK3/2T/13 Efficient Vocabularies Management Using Ontologies of the Czech Technical University in Prague.

References

1. Angele, J., Kifer, M., Lausen, G.: Ontologies in F-Logic. In: Staab, S., Studer, R. (eds.) Handbook on Ontologies, pp. 45–70. Springer Berlin Heidelberg (2009). https://doi.org/10.1007/978-3-540-92673-3_2
2. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, New York (2003)
3. Balaban, M.: The F-logic Approach for Description Languages. Annals of Mathematics and Artificial Intelligence **15**(1), 19–60 (1995). https://doi.org/10.1007/BF01535840
4. Booch, G.: Object-oriented Analysis and Design with Applications (2Nd Ed.). Benjamin-Cummings Publishing Co., Inc., Redwood City, CA, USA (1994)
5. Broekstra, J., Kampman, A., van Harmelen, F.: Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. In: Proceedings of the First International Semantic Web Conference on The Semantic Web. pp. 54–68 (2002)
6. de Bruijn, J., Heymans, S.: Translating ontologies from predicate-based to frame-based languages. In: Proceedings of the 2nd International Conference on Rules and Rule Markup Languages for the Semantic Web (RuleML2006) (2006)
7. de Bruijn, J., Lara, R., Polleres, A., Fensel, D.: OWL DL vs. OWL Flight: Conceptual Modeling and Reasoning for the Semantic Web. In: Proceedings of the 14th International Conference on World Wide Web. WWW '05, ACM (2005). https://doi.org/10.1145/1060745.1060836
8. Carroll, J.J., Dickinson, I., Dollin, C., Reynolds, D., Seaborne, A., Wilkinson, K.: Jena: Implementing the Semantic Web Recommendations. In: Proceedings of the 13th international World Wide Web conference (Alternate Track Papers & Posters). pp. 74–83 (2004)
9. Damásio, C.V., Analyti, A., Antoniou, G., Wagner, G.: Supporting Open and Closed World Reasoning on the Web. In: Alferes, J.J., Bailey, J., May, W., Schwertel, U. (eds.) Principles and Practice of Semantic Web Reasoning. pp. 149–163. Springer Berlin Heidelberg, Berlin, Heidelberg (2006). https://doi.org/10.1007/11853107_11
10. Donini, F.M., Nardi, D., Rosati, R.: Description Logics of Minimal Knowledge and Negation As Failure. ACM Trans. Comput. Logic **3**(2), 177–225 (Apr 2002). https://doi.org/10.1145/505372.505373
11. Eiter, T., Ianni, G., Lukasiewicz, T., Schindlauer, R., Tompits, H.: Combining answer set programming with description logics for the Semantic Web. IEEE Transactions on Knowledge and Data Engineering **22**(11), 1577–1592 (2010). https://doi.org/10.1109/TKDE.2010.111
12. Grosof, B.N., Horrocks, I., Volz, R., Decker, S.: Description logic programs: Combining logic programs with description logic. In: Proceedings of the 12th International Conference on World Wide Web. pp. 48–57. WWW '03, ACM, New York, NY, USA (2003). https://doi.org/10.1145/775152.775160
13. Grove, M.: Empire: RDF & SPARQL Meet JPA. semanticweb.com (April 2010), `http://semanticweb.com/empire-rdf-sparql-meet-jpa_b15617`

14. Horridge, M., Bechhofer, S.: The OWL API: A Java API for OWL ontologies. Semantic Web – Interoperability, Usability, Applicability (2011)
15. Horrocks, I., Kutz, O., Sattler, U.: The Even More Irresistible $\mathcal{SROIQ}$. In: Proc. of the 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2006). pp. 57–67 (2006)
16. Kattenstroth, H., May, W., Schenk, F.: Combining OWL with F-Logic Rules and Defaults. In: Proceedings of the ICLP'07 Workshop on Applications of Logic Programming to the Web, Semantic Web and Semantic Web Services, ALPSWS 2007 (2007)
17. Kifer, M.: Rules and Ontologies in F-Logic. In: Eisinger, N., Małuszyński, J. (eds.) Reasoning Web: First International Summer School 2005, Msida, Malta, July 25-29, 2005, Revised Lectures, pp. 22–34. Springer Berlin Heidelberg (2005). https://doi.org/10.1007/11526988_2
18. Kifer, M., Lausen, G., Wu, J.: Logical foundations of object-oriented and frame-based languages. J. ACM **42**(4), 741–843 (Jul 1995). https://doi.org/10.1145/210332.210335
19. Knorr, M., Alferes, J.J., Hitzler, P.: Local closed world reasoning with description logics under the well-founded semantics. Artificial Intelligence **175**(9-10), 1528–1554 (2011). https://doi.org/10.1016/j.artint.2011.01.007, `http://dx.doi.org/10.1016/j.artint.2011.01.007`
20. Křemen, P.: Building Ontology-Based Information Systems. Ph.D. thesis, Czech Technical University, Prague (2012)
21. Křemen, P., Kouba, Z.: Ontology-Driven Information System Design. IEEE Transactions on Systems, Man, and Cybernetics: Part C **42**(3), 334–344 (May 2012)
22. Ledvinka, M., Křemen, P.: A comparison of object-triple mapping libraries. Semantic Web p. 43 (feb 2019). https://doi.org/10.3233/SW-190345
23. Lloyd, J.W.: Foundations of Logic Programming. Springer-Verlag, Berlin, Heidelberg (1984)
24. Motik, B., Horrocks, I., Sattler, U.: Bridging the gap between OWL and relational databases. Web Semantics: Science, Services and Agents on the World Wide Web **7**(2) (2009)
25. Oren, E., Heitmann, B., Decker, S.: ActiveRDF: Embedding SemanticWeb data into object-oriented languages. Web Semantics: Science, Services and Agents on the World Wide Web **6**(3) (2008)
26. Patel-Schneider, P.F., Franconi, E.: Ontology Constraints in Incomplete and Complete Data. In: Proceedings of the 11th International Conference on The Semantic Web - Volume Part I. pp. 444–459. ISWC'12, Springer-Verlag, Berlin, Heidelberg (2012). https://doi.org/10.1007/978-3-642-35176-1_28
27. Sengupta, K., Krisnadhi, A.A., Hitzler, P.: Local closed world semantics: Grounded circumscription for OWL. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **7031 LNCS**(PART 1), 617–632 (2011). https://doi.org/10.1007/978-3-642-25073-6_39
28. Tao, J., Sirin, E., Bao, J., McGuinness, D.L.: Integrity Constraints in OWL. In: Fox, M., Poole, D. (eds.) AAAI. AAAI Press (2010)
29. Völkel, M., Sure, Y.: RDFReactor - From Ontologies to Programmatic Data Access. In: Poster and Demo at International Semantic Web Conference (ISWC) 2005, Galway, Ireland (2005), `https://pdfs.semanticscholar.org/40fa/7f7ebf5e3be92e6d3c9bad7b2753e9121971.pdf`, accessed 2018-12-28
30. Wenzel, K.: KOMMA: An Application Framework for Ontology-based Software Systems. Semantic Web – Interoperability, Usability, Applicability (2010)

31. Yang, G., Kifer, M.: Reasoning about Anonymous Resources and Meta Statements on the Semantic Web, pp. 69–97. Springer Berlin Heidelberg, Berlin, Heidelberg (2003). https://doi.org/10.1007/978-3-540-39733-5_4