Ontology Constraints in Incomplete and Complete Data

Peter F. Patel-Schneider and Enrico Franconi

Free University of Bozen-Bolzano, Italy
pfpschneider@gmail.com, franconi@inf.unbz.it

Abstract. Ontology and other logical languages are built around the idea that axioms enable the inference of new facts about the available data. In some circumstances, however, the data is meant to be complete in certain ways, and deducing new facts may be undesirable. Previous approaches to this issue have relied on syntactically specifying certain axioms as constraints or adding in new constructs for constraints, and providing a different or extended meaning for constraints that reduces or eliminates their ability to infer new facts without requiring the data to be complete. We propose to instead directly state that the extension of certain concepts and roles are complete by making them DBox predicates, which eliminates the distinction between regular axioms and constraints for these concepts and roles. This proposal eliminates the need for special semantics and avoids problems of previous proposals.

1 Introduction

A complaint against ontology languages like the W3C OWL Web Ontology Language is that axioms in such ontology languages sometimes have too many consequences. For example,[1] the axiom $\mathsf{MarriedPerson} \sqsubseteq \forall\mathsf{hasSpouse}.\mathsf{Person}$ along with $\mathsf{MarriedPerson}(\mathsf{Peter})$ and $\mathsf{hasSpouse}(\mathsf{Peter}, \mathsf{Susan})$ produces the consequence $\mathsf{Person}(\mathsf{Susan})$. Similarly, the axiom $\mathsf{MarriedPerson} \equiv \mathsf{Person} \sqcap (= 1\,\mathsf{hasSpouse})$ along with $\mathsf{MarriedPerson}(\mathsf{Peter})$ implies that there is some spouse for Peter. However, the data generation methodology may be such that all persons and/or all spouse relationships should be explicitly given, and not inferred. For example, the data for these concepts or roles may come from an ostensibly complete database. If this is the case, then problems arise when using ontology axioms in the usual way. In the first example Susan is inferred to be a person even if there may be no explicit fact to this effect (and thus Susan is not a person) and in the second example some (unknown) spouse is inferred for Peter, which is not an explicit fact at all.

This particular complaint against standard ontology languages has spawned a number of proposals attempting to overcome the problem. Some proposals [10,15] change certain axioms into integrity constraints (roughly statements that

[1] Throughout this paper a standard concise syntax [1] for OWL and other ontology languages and description logics will be used in examples to improve readability.

check the integrity of the data in the knowledge base (KB) instead of enabling consequences) that are interpreted in minimal or modal ways. Other proposals [6] suggest that the ontology language be extended with autoepistemic constructs that can be used to write integrity constraints.

We disagree with the general approach taken in all of these proposals. Our belief is that the problem here is not a deficiency in ontology languages at all. In particular, we do not find any problems with the ontology axioms above, nor with their consequences. To the contrary, these axioms are unexceptional and should infer *consequences* just like any other axiom. Instead we claim that the complaints against the power of axioms in ontology languages have to do with a mismatch between the general open-world assumption in ontology languages and in many other logics, on one hand, and a desire to have complete and maybe even explicit information about the extension of certain concepts and roles, on the other.

So, for example, if we have complete *data* about some concepts or roles, e.g., a set of the distinct instances of Person or the distinct pairs in hasSpouse, then it should be the case that inferences *do not* augment this data. But, again, this is not a problem with the axioms, e.g., the ones above, which are simply a reflection of the way we believe the domain is, but is instead a problem with our data, which is incomplete when it was stated to be complete. A consequence that adds more information to these concepts or roles then is contradicting this supposed completeness, causing an inconsistency in the KB.

Our basic approach to modeling, then, is to build the ontology without considering anything like the desired completeness of data or integrity constraints. Only then do we determine which concepts and roles are to have complete data. So, for example, we might build the ontology of people and spouses

$$\begin{aligned}
&\mathsf{Person} \sqsubseteq \top\\
&\mathsf{MarriedPerson} \equiv \mathsf{Person} \sqcap (= 1\ \mathsf{hasSpouse})\\
&\mathsf{MarriedPerson} \sqsubseteq \forall \mathsf{hasSpouse}.\mathsf{Person}
\end{aligned}$$

We might, then, perhaps because we are using an explicit data source of all spousal relationships, require that the hasSpouse role be completely and explicitly provided.

Suppose that data in the hasSpouse role is provided as the four distinct entries

hasSpouse

Peter	Susan
Susan	Peter
Mary	Paul
Paul	Mary

(1)

Our ontology axioms will make inferences from this data, including Person(Susan) and MarriedPerson(Peter) which is all as it should be. Suppose, however, that our KB also includes MarriedPerson(Alex) Then our ontology axioms would, in effect, add another entry to the hasSpouse role, one with first entry Alex. As we stated that the given data for hasSpouse is complete, this would be an inconsistent situation indicating that the constraint has been violated.

This treatment of complete concepts and roles is just the DBox treatment [14,7]. Our approach has all the benefits of DBoxes, including easy query answering when all concepts and roles are DBox concepts and roles, and *exact answers* to queries controlled by DBox concepts and roles, i.e., if we query for married people and their spouses above we get precisely every spousal relationship. These exact answers can be used in applications without worrying that there might be missing information, i.e., as might happen when there is a married person whose spouse is not known. Applications can treat such query answers in the same way that they can treat query answers in databases.

2 Autoepistemic and Minimal Model Approaches

Autoepistemic extensions to description logics or ontology languages, such as the work by Donini *et al* on description logics of minimal knowledge and negation as failure (MKNF-DL) [6], can be used to express constraints, either directly, or as rules [12]. Negation as failure by itself has been used to express constraints in OWL Flight [5,4].

The MKNF-DL axiom

$$\mathbf{K}\mathsf{MarriedPerson} \sqsubseteq \exists \mathbf{A}\mathsf{hasSpouse} \tag{2}$$

expresses the constraint that every known married person has a known spouse so that a KB containing only

$$\begin{array}{l} \mathbf{K}\mathsf{MarriedPerson} \sqsubseteq \exists \mathbf{A}\mathsf{hasSpouse} \\ \mathsf{MarriedPerson}(\mathsf{Joe}) \end{array} \tag{2.1}$$

is unsatisfiable because Joe does not have a known spouse. Knowledge in MKNF-DL requires knowing what, not just knowing that, so

$$\begin{array}{ll} \mathbf{K}\mathsf{MarriedPerson} \sqsubseteq \exists \mathbf{A}\mathsf{hasSpouse} & \\ \mathsf{MarriedPerson}(\mathsf{Joe}) & (\exists \mathsf{hasSpouse})(\mathsf{Joe}) \end{array} \tag{2.2}$$

is also unsatisfiable because, although Joe does have some spouse, the identity of that spouse is not known. Only information about the identity of Joe's spouse is adequate to satisfy the constraint, as in

$$\begin{array}{ll} \mathbf{K}\mathsf{MarriedPerson} \sqsubseteq \exists \mathbf{A}\mathsf{hasSpouse} & \\ \mathsf{MarriedPerson}(\mathsf{Joe}) & \mathsf{hasSpouse}(\mathsf{Joe}, \mathsf{Susan}) \end{array} \tag{2.3}$$

Adding autoepistemic constructs into description logics augments their expressive power considerably. Although the result is decidable, we are unaware of any high-performance systems that implement reasoning in MKNF-DL.

The constraint axioms in the MKNF-DL approach involve autoepistemic operators in negative contexts, such as $\mathbf{K}\mathsf{MarriedPerson}$ above, indicating that lack of knowledge can allow a constraint to be satisfied. Constraint axioms thus do not necessarily imply their non-constraint version, so one often needs to provide two versions of constraint axioms, such as by adding

$$\mathsf{MarriedPerson} \sqsubseteq {=}1\ \mathsf{hasSpouse}$$

Without such an axiom, as in KB 2.1, it will not be the case that all married-Persons have a spouse, only the known ones.

For similar reasons, the MKNF-DL approach does not require that existential individuals be considered in constraints. For example, in

$$\begin{aligned}&\mathbf{K}\mathsf{MarriedPerson} \sqsubseteq \exists \mathbf{A}\mathsf{hasSpouse}\\&(\exists \mathsf{hasChild.MarriedPerson})(\mathsf{Mary})\end{aligned} \tag{2.4}$$

the constraint is satisfied even though nothing is known about Mary's married child. We feel that constraints should be active on all participating individuals, independently of whether they are known, and consider that the example above points out a major problem with MKNF-DL and similar modal approaches.

Also similarly, the MKNF-DL approach does not require that disjunctive information be considered by constraints. So

$$\begin{aligned}&\mathsf{Student} \equiv \mathsf{UGStudent} \sqcup \mathsf{GStudent}\\&\mathbf{K}\mathsf{UGStudent} \sqsubseteq \exists \mathbf{A}\mathsf{major}\\&\mathbf{K}\mathsf{GStudent} \sqsubseteq \exists \mathbf{A}\mathsf{faculty}\\&\mathsf{Student}(\mathsf{Mary})\end{aligned} \tag{3}$$

is satisfiable even though Mary must be either an undergraduate, in which case the constraint about majors is not satisfied, or a graduate, in which the constraint about faculties is not satisfied. We feel that constraints should take into account such disjunctive information and consider that this example points out another major problem with this kind of approach.

Motik *et al* [10] have proposed a very different approach to integrity constraints. Instead of extending the language itself, they divide up axioms into three categories. As is usual, facts are segregated into the ABox ($\mathcal{A}$). Other axioms, however, are divided between regular axioms (the standard TBox or $\mathcal{S}$) and constraints ($\mathcal{C}$). Next, minimal Herbrand models are defined. In this approach a minimal Herbrand model is a Herbrand model where the extension of all predicates, even equivalence (the predicate standing in for equality), is minimized. A constraint is then satisfied by a standard TBox and an ABox if all minimal Herbrand models of the standard TBox plus the ABox are also models of the constraint. Because each minimal model of the standard TBox plus the ABox is a model of the standard TBox, it is obvious that in this approach any constraint entailed by the standard TBox is satisfied in the KB.

In the KB

$$\begin{aligned}&\mathcal{C} : \mathsf{MarriedPerson} \sqsubseteq \exists \mathsf{hasSpouse}\\&\mathcal{A} : \mathsf{MarriedPerson}(\mathsf{Joe})\end{aligned} \tag{2.1_m}$$

the first axiom is considered as a constraint and the second as a fact in the ABox. The constraint is not satisfied in this KB because in the (only) minimal model of $\mathcal{A}$ Joe has no spouse.

The minimal model approach differs considerably from the MKNF-DL approach. For example in

$$\begin{array}{ll} \mathcal{S} : \mathsf{MarriedPerson} \sqsubseteq \exists \mathsf{hasSpouse} & \\ \mathcal{C} : \mathsf{MarriedPerson} \sqsubseteq \exists \mathsf{hasSpouse} & \qquad (2.2_m) \\ \mathcal{A} : \mathsf{MarriedPerson}(\mathsf{Joe}) & \end{array}$$

the constraint is satisfied, even though there is no known spouse for Joe, because in each minimal model, Joe has a spouse. We consider this ability to utilize "unknown", or existential, fillers to fulfill constraints as a problem with this approach.

The particular problem with existentials in the previous example can be alleviated by introducing an extra predicate (O) that is asserted true of every identifier in the KB, as in

$$\begin{array}{lll} \mathcal{S} : \mathsf{MarriedPerson} \sqsubseteq \exists \mathsf{hasSpouse} & & \\ \mathcal{C} : \mathsf{MarriedPerson} \sqsubseteq \exists \mathsf{hasSpouse} \sqcap \mathsf{O} & & \qquad (2.2'_m) \\ \mathcal{A} : \mathsf{MarriedPerson}(\mathsf{Joe}) & \mathsf{O}(\mathsf{Joe}) & \end{array}$$

Here Joe's spouse is not in O so the constraint is not satisfied. One can think of O as holding the requirement of having a name.

However, other problems with this minimal model approach cannot be overcome. If a filler can be one of two possibilities (both named) then a constraint requiring a filler will be satisfied, even though the identity of the filler is not known. For example, in

$$\begin{array}{lll} \mathcal{S} : \mathsf{JoeClass} \sqsubseteq \exists \mathsf{hasSpouse}.\{\mathsf{Mary}, \mathsf{Susan}\} & & \\ \mathcal{C} : \mathsf{MarriedPerson} \sqsubseteq \exists \mathsf{hasSpouse} & & \qquad (2.5_m) \\ \mathcal{A} : \mathsf{MarriedPerson}(\mathsf{Joe}) & \mathsf{JoeClass}(\mathsf{Joe}) & \end{array}$$

the constraint is satisfied even though Joe's spouse is not known. We consider this to be a major problem with this approach, as we view the goal of integrity constraints to be checking data, not checking possibilities.

Another problem with the minimal models approach is that all concepts and roles are minimized. This means that the presence of axioms that cause one concept or role to grow when something else shrinks disturbs the minimization in unusual ways. For example, the constraint in

$$\begin{array}{llll} \mathcal{C} : \mathsf{RParent} \sqsubseteq \geqslant 2\, \mathsf{hasChild} & & & \qquad (4) \\ \mathcal{A} : \mathsf{RParent}(\mathsf{Joe}) & \mathsf{hasChild}(\mathsf{Joe}, \mathsf{Mary}) & \mathsf{hasChild}(\mathsf{Joe}, \mathsf{Susan}) & \end{array}$$

is satisfied, as expected, because the minimal model minimizes away the possible equality between Mary and Susan. However, extending the KB by adding a definition, as in

$$\begin{aligned} &\mathcal{S} : \mathsf{DParent} \equiv \geqslant 2\, \mathsf{hasChild} \\ &\mathcal{C} : \mathsf{RParent} \sqsubseteq \geqslant 2\, \mathsf{hasChild} \\ &\mathcal{A} : \mathsf{RParent(Joe)} \qquad \mathsf{hasChild(Joe, Mary)} \qquad \mathsf{hasChild(Joe, Susan)} \end{aligned} \tag{4.1}$$

can make the constraint not be satisfied. The reason for this *unexpected* result is that DParent grows when equality shrinks. This results in a minimal model where Mary and Susan are the same and Joe is not in DParent. In this minimal model Joe has only one child and this violates the constraint. We view this as a very serious problem with the approach, particularly as facts and queries in the approach have to use atomic concepts and roles, and thus may require the introduction of extra predicates.

A major reason for the difference between the MKNF-DL approach and the minimal model approach has to do with the modal (or non-modal) nature of the approaches. The modal parts of constraints in MKNF-DL consider what is known to be true in all models, so **K**UGStudent picks out named individuals who are known to be graduate students in all models. In the minimal model approach constraints are directly evaluated in each minimal model, so there is no consideration of the situation in other models.

A third approach [15] to integrity constraints combines axiom segregation and equality minimization, somewhat as in the minimal model approach, with a portion of the autoepistemic nature of the MKNF-DL approach. In this hybrid approach, there is a two-way division in extended KBs between $\mathcal{K}$, the regular KB ($\mathcal{T}$ and $\mathcal{A}$ of the previous approach), and $\mathcal{S}$, the constraints, both of which can contain both axioms and facts. The minimal equality models of a KB are defined as the regular models of the KB that are minimal with respect to equality between individuals names, with all else remaining fixed. Constraints are interpreted in a modal setting where atomic concepts and roles are interpreted, roughly, as names (pairs of names) belonging to the concept (role) in all minimal models of the KB.

One might think that only minimizing equality avoids the problems in the previous approach with respect to additional predicates. Unfortunately, this is not the case. For example, the extended KB

$$\begin{aligned} &\mathcal{C} : \mathsf{RParent} \sqsubseteq \geqslant 2\, \mathsf{hasChild} \\ &\mathcal{T} : \mathsf{RParent(Joe)} \qquad \mathsf{hasChild(Joe, Mary)} \qquad \mathsf{hasChild(Joe, Susan)} \end{aligned} \tag{4_h}$$

is valid, as expected, but the extended extended KB

$$\begin{aligned} &\mathcal{C} : \mathsf{RParent} \sqsubseteq \geqslant 2\, \mathsf{hasChild} \\ &\mathcal{T} : \mathsf{DParent} \equiv \geqslant 2\, \mathsf{hasChild} \\ &\quad\ \ \mathsf{RParent(Joe)} \qquad \mathsf{hasChild(Joe, Mary)} \qquad \mathsf{hasChild(Joe, Susan)} \end{aligned} \tag{4.1_h}$$

is *not*. Here the presence of DParent prevents minimizing away the possible equality between Mary and Susan, because making Mary and Susan the same causes a change in the extension of DParent. As is the case for the previous approach, we view this fragility of minimization as a very serious problem.

This combination approach also suffers from variants of many of the problems of the MKNF-DB approach, including both extended KBs

$$\begin{aligned}&\mathcal{C} : \mathsf{Child} \sqsubseteq \bot \\ &\mathcal{T} : (\exists \mathsf{hasSpouse}.\mathsf{Spouse})(\mathsf{Joe})\end{aligned} \tag{5.1}$$

and

$$\begin{aligned}&\mathcal{C} : \mathsf{Child} \sqsubseteq \bot \\ &\mathcal{T} : (\forall \mathsf{hasSpouse}.\mathsf{Child})(\mathsf{Joe}) \qquad (\exists \mathsf{hasSpouse}.\{\mathsf{Mary}, \mathsf{Susan}\})(\mathsf{Joe})\end{aligned} \tag{5.2}$$

being valid. However, the situation is even worse here, as one might argue that the MKNF-DB construct analogous to the constraint, $\mathsf{Child} \sqsubseteq \bot$, should not be considered to be a constraint, but such arguments cannot be made when constraints are explicitly given.

Local closed world semantics has a relationship to integrity constraints, as can be seen from the fact that all of the above approaches employ some form of minimization. A proposal to add local closed world semantics to OWL [13] uses grounded circumscription to avoid undecidability problems with circumscription. Grounded circumscription is just regular circumscription, except that minimized concepts (roles) can only contain named individuals (pairs of named individuals).

Grounded circumscription can capture some common aspects of integrity constraints. The basic idea is to evaluate the constraints after grounded circumscription has been applied. The advantage of circumscription over other minimization methods is that only certain predicates are minimized, while other are fixed or allowed to vary. In this way it might be possible to alleviate (but probably not completely overcome) the problems of simpler minimization methods.

A major problem with this approach, however, is the difficulty of performing even grounded circumscription. When circumscribing, one has to guess which named individuals (pairs of named individuals) are in each minimized concept (role), and only then determine whether the guess is acceptable. Then the minimal acceptable guesses become the actual minimizations. When the KB is even of only a moderate size, this can take an extremely long time.

3 Constraints with Complete Information

Our proposal does not depend on any of these modal or minimal model techniques to prevent constraints from enabling inferences. Instead, as stated earlier, we specify that certain concepts and roles have complete information. Then for these concepts and roles no information can be added, turning axioms into constraints for them. In effect, axioms can only check that information is already in the complete concept or role, precisely as is wanted for integrity constraints.

If an axiom plus some data produces a consequence that adds information to a complete concept or role, then an inconsistency results. What happens after an inconsistency is detected is outside the scope of the logic. Generally, some modification would be needed to the assertions in the KB, which might involve removing an assertion that enabled the attempt to add the offending information. The modification might, on the other hand, actually be to change the information in the complete concept or role, but this would occur as a step outside of the logic.

3.1 OWL and $\mathcal{SROIQ}(\mathcal{D})$

As much of our proposal is related to the W3C OWL Web Ontology Language [11] and related ontology languages, we introduce OWL, via $\mathcal{SROIQ}(\mathcal{D})$ [9], the description logic underlying OWL.

Let $\mathbf{C}$ be a set of *concept names*, $\mathbf{D}$ be a set of *datatype names*, $\mathbf{R}$ be a set of *abstract role names*, $\mathbf{T}$ be a set of *concrete role names*, $\mathbf{I}$ be a set of *individual names*, and $\mathbf{V}$ be a set of *data values*, with $\mathbf{C} \cap \mathbf{D} = \phi$, $\mathbf{R} \cap \mathbf{T} = \phi$, and $\mathbf{I} \cap \mathbf{V} = \phi$.

$\mathcal{SROIQ}(\mathcal{D})$ concepts (C), datatypes (D), and abstract roles (R) are constructed via

$$\begin{aligned} C ::= & \top \mid A \mid \{a\} \mid \neg C \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid \exists R.C \mid \forall R.D \mid \exists R.Self \mid \\ & \geqslant n\, R.C \mid \leqslant n\, R.C \mid \exists T.D \mid \forall T.D \mid \geqslant n\, T.D \mid \leqslant n\, T.D \\ D ::= & B \mid \{v\} \mid \neg D \mid D_1 \sqcap D_2 \mid B \leq v \mid B < v \mid B > v \mid B \geq v \\ R ::= & \top \mid P \mid P^- \end{aligned}$$

where $A \in \mathbf{C}$, $B \in \mathbf{D}$, $P \in \mathbf{R}$, $T \in \mathbf{T}$, $a \in \mathbf{I}$, $v \in \mathbf{V}$, and n is a non-negative integer.

A general concept inclusion axiom (GCI) is $C_1 \sqsubseteq C_2$, for C_1, C_2 both concepts. A role inclusion axiom (RIA) is $R_1 \circ \ldots \circ R_n \sqsubseteq R$, for R, R_i all roles or $T_1 \sqsubseteq T$, for $T, T_1 \in \mathbf{T}$.

A role assertion is *Sym*(R), *Tra*(R), *Ref*(R), *Irr*(R), or *Dis*(R_1, R_2), for R, R_1, R_2 any role except $\top$; or $\mathsf{Dis}(T_1, T_2)$, for $T_1, T_2 \in \mathbf{T}$. An individual assertion is $C(a)$, $R(a_1, a_2)$, $(\neg R)(a_1, a_2)$, $T(a, v)$, $a_1 = a_2$, or $a_1 \neq a_2$, for C a concept, R a role, $T \in \mathbf{T}$, $a, a_1, a_2 \in \mathbf{I}$, and $v \in \mathbf{V}$.

A $\mathcal{SROIQ}(\mathcal{D})$ KB is a pair $\langle \mathcal{T}, \mathcal{A} \rangle$. $\mathcal{T}$ (the TBox) is a finite set of GCIs and RIAs and role assertions such that the RIAs in $\mathcal{T}$ form a regular role hierarchy (see [9]), the role assertions in $\mathcal{T}$ are simple in $\mathcal{T}$, and each role in an $\geqslant R C$. or $\exists R.Self$ or $\mathsf{Irr}(R)$ or $\mathsf{Dis}(R_1, R_2)$ is simple in $\mathcal{T}$ (see [9]). $\mathcal{A}$ (the ABox) is a finite set of individual assertions such that each role, R, in a $(\neg R)(a, b)$ is simple in $\mathcal{T}$. The names and values of the KB form its signature, $\mathcal{S}$.

The semantics of $\mathcal{SROIQ}(\mathcal{D})$ are as for $\mathcal{SROIQ}$ [9] and as for the W3C OWL 2 Web Ontology Language [11]. Here we present only the general notions of their semantics. Much of our proposal works for any ontology language or description logic or even any fragment of first-order logic, so we will provide a a general semantic framework that is suitable for any of these languages.

Semantics are based on interpretations, $\mathcal{I}$, that map, $\cdot^{\mathcal{I}}$, constants (individual names and data values) into elements of a domain, $\Delta^{\mathcal{I}}$, concept and datatype names into subsets of the domain, and abstract and concrete role names into sets of pairs over the domain. Data values and datatype names have fixed mapping.

This mapping is extended to all syntactic constructs in the language, mapping closed formulae (axioms and assertions) into either true or false (satisfying or not satisfying them, respectively). An interpretation is a model of a KB consisting of a finite set of closed formulae (axioms and assertions), written $\mathcal{I} \models \mathcal{KB}$, iff it satisfies all the formulae the KB.

3.2 DBoxes

Given a $\mathcal{SROIQ}(\mathcal{D})$ (or other description logic) KB, a DBox [14], $\mathcal{DB}$, is a finite set of atomic individual assertions of the form $A(a)$ or $P(a_1, a_2)$ or $T(a, v)$ for $A \in \mathbf{C}$, $P \in \mathbf{R}$, $T \in \mathbf{T}$, $a, a_1, a_2 \in \mathbf{I}$, and $v \in \mathbf{V}$. The signature of a DBox $\mathcal{DB}$, called $\mathcal{S}(\mathcal{DB})$, contains all the concept, abstract or concrete role names occurring in $\mathcal{DB}$. A $\mathcal{SROIQ}(\mathcal{D})$ (or other description logic) KB plus DBox is a triple $\langle \mathcal{T}, \mathcal{A}, \mathcal{DB} \rangle$ where $\mathcal{T}$ and $\mathcal{A}$ are as before and $\mathcal{DB}$ is a DBox. The active domain of a DBox $\mathcal{DB}$, $\mathbf{I}_{\mathcal{DB}} \subseteq \mathbf{I}$, is the set of all individuals appearing in the DBox. An interpretation, $\mathcal{I}$, of $\langle \mathcal{T}, \mathcal{A}, \mathcal{DB} \rangle$ is just an interpretation of $\langle \mathcal{T}, \mathcal{A} \rangle$ plus

- for each individual name $a \in \mathbf{I}_{\mathcal{DB}}$: $a^{\mathcal{I}} = a$, (i.e., the standard name assumption for DBox individuals);
- for each pair of distinct individual names $a_1 \in \mathbf{I}_{\mathcal{DB}}$ and $a_2 \in \mathbf{I}$: $a_1^{\mathcal{I}} \neq a_2^{\mathcal{I}}$, (i.e., the unique name conditions for DBox individuals);
- for each concept name $A \in \mathcal{S}(\mathcal{DB})$, $\quad x \in A^{\mathcal{I}}$ iff $\exists A(a) \in \mathcal{DB} \ : \ a^{\mathcal{I}} = x$;
- for each abstract role name, $P \in \mathcal{S}(\mathcal{DB})$,
  $\langle x, y \rangle \in P^{\mathcal{I}}$ iff $\exists P(a_1, a_2) \in \mathcal{DB} \ : \ a_1^{\mathcal{I}} = x \wedge a_2^{\mathcal{I}} = y$; and
- for each concrete role name, $T \in \mathcal{S}(\mathcal{DB})$,
  $\langle x, y \rangle \in T^{\mathcal{I}}$ iff $\exists T(a, v) \in \mathcal{DB} \ : \ a^{\mathcal{I}} = x \wedge v^{\mathcal{I}} = y$.

The essence of a DBox is that the extension of each concept or role that shows up in the DBox is completely determined by the DBox, much as it would be by a database table. This requires the standard name assumption in the DBox. To emphasize the relationship between database tables and DBox concept and roles, we will often write the assertions for the concept or role in tabular form, as in the case of the hasSpouse role in the KB (1) above.

It can been shown (see [8]) that it is harmless to drop the standard name assumption for DBox individuals, in presence of the unique name conditions for DBox individuals: the spurious models in the weaker KB are indistinguishable from the good ones. It is also possible to fully encode a KB with a DBox into an equivalent KB in an expressive description logic such as $\mathcal{SROIQ}(\mathcal{D})$ or OWL (see [7]). The unique name conditions for DBox individuals can be easily written as a finite set of individual inequality assertions. To rewrite a DBox concept, C, with $C(i_1), \ldots, C(i_n)$ in the DBox, simply add the DBox assertions for C to the ABox and add $C \sqsubseteq \{i_1, \ldots, i_n\}$ to the TBox. To rewrite a DBox abstract or concrete role, R, with $R(i_1, v_1), \ldots, R(i_n, v_n)$ in the DBox, simply add the DBox assertions for R to the ABox, add $\exists R \sqsubseteq \{i_1, \ldots, i_n\}$ to the TBox, and, for each i_j, $1 \leq j \leq n$, add $(\forall R.\{v_{j_1}, \ldots, v_{j_{m_k}}\})(i_j)$ to the ABox, where $\{v_{j_1}, \ldots, v_{j_{m_k}}\}$ is the set of R-fillers for i_j in the DBox. As a consequence of this easy polynomial embedding, we can conclude that reasoning with DBoxes in such expressive ontology languages is not harder than classical reasoning without DBoxes, and it can be implemented without changing anything with respect to the classical case.

3.3 Completely Specified Concepts and Roles

We introduce in this section the definition of concepts and roles *completely specified* from DBox predicates, a notion strictly related to *determinacy* and *implicit definability* [2,8].

Definition (Completely Specified Concept or Role). *Let $\mathcal{I}$ and $\mathcal{J}$ be any two models of a KB plus DBox $\langle \mathcal{T}, \mathcal{A}, \mathcal{DB} \rangle$. A concept (resp. role) C (resp. R) is* completely specified *(or* determined*) by the DBox predicates $\mathcal{S}(\mathcal{DB})$ in the KB if and only if whenever I and J agree on the interpretation given to each concept and role in $\mathcal{S}(\mathcal{DB})$ then $C^{\mathcal{I}} = C^{J}$ (resp. $R^{\mathcal{I}} = R^{J}$).*

This definition states that a concept C (resp., a role R) is completely specified by the DBox predicates in a KB if and only if all models of the KB that interpret the symbols in $\mathcal{S}_{\mathbf{DB}}$ the same way also keep the interpretation for C (resp., R) fixed. In other words, once a DBox is fixed (and therefore the interpretation of all the DBox predicates is always the same in any interpretation) then also the interpretation of the completely specified predicates (concepts and/or roles) is fixed. It is as if the concept or role augments the original DBox with its own extension.

It is obvious that a DBox concept or role is completely specified by the DBox predicates, given the close correspondence between the definition of DBoxes and complete specification. It is also possible to completely specify a concept or role in other ways. For example, a concept or role might be defined to be equivalent to a DBox concept or role. Other, more complex, definitions can also completely specify a concept or role in terms of DBox concepts or roles. For example, from KBs of the following form it is possible to derive that $\mathsf{GStudent}$ is a completely specified concept, given that the concepts $\mathsf{Student}$ and $\mathsf{UGStudent}$ are DBox concepts:

$$\begin{aligned}
\mathcal{T} : \ & \mathsf{Student} \sqsubseteq \mathsf{UGStudent} \sqcup \mathsf{GStudent} \\
& \mathsf{UGStudent} \sqsubseteq \neg\mathsf{GStudent} \sqcap \mathsf{Student} \\
& \mathsf{GStudent} \sqsubseteq \mathsf{Student} \\
\mathcal{S}(\mathcal{DB}) : \ & \{\mathsf{Student}, \mathsf{UGStudent}\}
\end{aligned} \tag{6}$$

Since the above KB induces a partition of the concept $\mathsf{Student}$ between the concept $\mathsf{UGStudent}$ and the concept $\mathsf{GStudent}$, whenever two of these concepts are completely specified (e.g., they are DBox concepts) then also the third is necessarily completely specified.

It is possible to determine whether a concept or role is completely specified by a set of DBox predicates in a KB using only standard description logic inferences [14,8].

3.4 Constraints

As our proposal is quite different from the previous proposals for integrity constraints, we will provide examples covering the major use cases for integrity constraints and several variants of their variants.

The entire extension of completely specified concepts and roles is known by name. This makes it quite obvious that completely specified concepts and roles naturally enforce constraints concerning knowing the identity and type of role fillers. For example, if hasSpouse is a completely specified role, then any fillers of hasSpouse will be known by name. If this is the case, the axiom

$$\mathsf{MarriedPerson} \sqsubseteq {=}1\,\mathsf{hasSpouse} \tag{7}$$

will be true in a KB only if each married person has precisely one known spouse. So, the KB

$$\begin{aligned} \mathcal{T} &: \mathsf{MarriedPerson} \sqsubseteq {=}1\,\mathsf{hasSpouse.Person} \\ \mathcal{A} &: \mathsf{MarriedPerson(Joe)} \\ &\ \mathsf{MarriedPerson(Jack)} \\ \mathcal{DB} &: \quad \mathsf{hasSpouse} \end{aligned} \tag{8}$$

Jack	Elizabeth
Jack	Liz

is unsatisfiable because Joe has no spouse (because he is distinct from Jack) and Jack has both Elizabeth and Liz as spouses (and they are distinct from each other). The situation would be completely different if hasSpouse was not a DBox role, in which case Joe would have been inferred to have some spouse, and Elizabeth and Liz would have been inferred to be the same.

Let's see now an example of an entailed constraint. Consider the KB

$$\begin{aligned} \mathcal{T} &: \mathsf{MarriedPerson} \sqsubseteq {\geqslant}1\,\mathsf{hasSpouse.Person} \\ \mathcal{DB} &: \quad \mathsf{hasSpouse} \end{aligned} \tag{9}$$

Jack	Elizabeth

From the above, we can entail the statement

$$\mathsf{MarriedPerson} \sqsubseteq {=}1\,\mathsf{hasSpouse.Person}$$

which couldn't be derived if hasSpouse were not a completely specified role.

All entries in completely specified concepts and roles have to have a name, i.e., not be some unknown filler, eliminating one problem with minimal models approaches (KB 2.2_m). Nor is it possible for disjunctive information to be adequate, eliminating another problem with minimal models approaches. (KB 2.5_m). Here our approach has the desirable behavior of autoepistemic approaches, requiring known certain fillers. Similarly, there is no issue with whether unknown objects are considered by constraints, which causes problems for autoepistemic approaches (KB 2.4). Here our approach has the desirable behavior of autoepistemic approaches, with constraints effectively taking into account all possible interpretations, because there is only one.

In the previous example, it was not necessary that the spouse be known to be a person, because Person was not a completely specified concept. If however, all people are known, then Person would be a completely specified concept, and spouses of married people would need to be known as people. It may be the case, however, that not all people are known, only those that are spouses. In this situation, MarriedPerson would be a completely specified concept,

$$\begin{aligned}\mathcal{T}: \ & \mathsf{MarriedPerson} \sqsubseteq {=}1\, \mathsf{hasSpouse.Person} \\ & \mathsf{MarriedPerson} \sqsubseteq {=}1\, \mathsf{hasSpouse.MarriedPerson} \\ & \mathit{Sym}(\mathsf{hasSpouse}) \end{aligned} \qquad (10)$$

$\mathcal{DB}$: hasSpouse

Jack	Elizabeth
Elizabeth	Jack

MarriedPerson

Jack
Elizabeth

Here any married person has to have precisely one spouse that is also a married person, as well as precisely one spouse that is a person. The extension of the concepts spouse and MarriedPerson provide precisely one known spouse that is known to be a married person as well as precisely one spouse overall for both Jack and Elizabeth, satisfying the constraint portions of these axioms. Because Jack and Elizabeth are each other's only spouse, they are both are inferred to belong to Person as well.

The situation where only a portion of a concept or role is completely specified and thus causes constraint-like behavior, is quite natural. For example, neither all people nor their SSN's may be known, but all taxpayers and their SSN's are. The following KB captures this situation:

$$\begin{aligned}\mathcal{T}: \ & \mathsf{Person} \sqsubseteq ({\leqslant}1\, \mathsf{hasSSN}) \sqcap (\forall \mathsf{hasSSN.integer}) \\ & \mathsf{TaxPayer} \sqsubseteq (\mathsf{Person}) \sqcap ({=}1\, \mathsf{hasSSNTP.integer}) \\ & \mathsf{hasSSNTP} \sqsubseteq \mathsf{hasSSN} \\ \mathcal{A}: \ & \mathsf{Person(Jill)} \\ & \mathsf{Person(Susan)} \\ & \mathsf{hasSSN(Jill, 987654321)} \end{aligned} \qquad (11)$$

$\mathcal{DB}$: TaxPayer

Jack
...

hasSSNTP

Jack	123456789
...	...

Here it does not matter that Susan's SSN is not known, but Jack's must be, and so must that of all the other taxpayers.

Because there is no minimization involved in our approach, there is no problem with extra axioms modifying the satisfaction of constraints. As occurs in the minimal models approaches (KB 4 and variants). In

$$\begin{aligned}
\mathcal{T}:\ & \mathsf{MDPerson} \equiv \mathsf{Person} \sqcap \geqslant 2\, \mathsf{hasDependent.Child} \\
& \mathsf{FDPerson} \equiv \mathsf{Person} \sqcap \leqslant 2\, \mathsf{hasDependent} \\
\mathcal{A}:\ & \mathsf{MDPerson(Joe)} \\
& \mathsf{FDPerson(Jack)}
\end{aligned}$$

$\mathcal{DB}$: (12)

hasDependent

Joe	Mary
Joe	Susan
Jack	Bill
Jack	John
Jack	Thomas

Child

Mary
Susan
Bill
John
Thomas

Mary and Susan are distinct, and no axioms can affect this situation, so Joe has two suitable dependents and the constraint axiom for MDPerson is satisfied on Joe regardless of anything else in the KB. Similarly, Bill, John, and Thomas are distinct so the constraint axiom for FDPerson is violated on Jack.

No disjunctive information can infect completely specified concepts or roles. In KB (3) about students, it is most likely that all of UGStudent, GStudent, major, and faculty are desired to be completely specified. If no other information is added, as in

$$\begin{aligned}
\mathcal{T}:\ & \mathsf{Student} \equiv \mathsf{UGStudent} \sqcup \mathsf{GStudent} \\
& \mathsf{UGStudent} \sqsubseteq \exists \mathsf{major} \\
& \mathsf{GStudent} \sqsubseteq \exists \mathsf{faculty} \qquad (13) \\
\mathcal{A}:\ & \mathsf{Student(Mary)}
\end{aligned}$$

$\mathcal{DB}$: UGStudent GStudent major faculty

then the KB is inconsistent, as Mary is neither an undergraduate nor a graduate, violating the first axiom.

The KB cannot be consistent without having Mary's situation as an undergraduate or graduate be provided, and then the required information about either her major or faculty, as in

$$\begin{aligned}
\mathcal{T}:\ & \mathsf{Student} \equiv \mathsf{UGStudent} \sqcup \mathsf{GStudent} \\
& \mathsf{UGStudent} \sqsubseteq \exists \mathsf{major} \\
& \mathsf{GStudent} \sqsubseteq \exists \mathsf{faculty} \qquad (14) \\
\mathcal{A}:\ & \mathsf{Student(Mary)}
\end{aligned}$$

$\mathcal{DB}$:

UGStudent

Mary

GStudent

major

Mary	Psychology

faculty

Our approach can naturally handle disjunctive information that interacts with completely specified concepts and roles. For example, if KB (13) is modified so that major and faculty are completely specified (admittedly not a very normal setup) as follows:

$$\begin{aligned}\mathcal{T} :\ & \mathsf{Student} \equiv \mathsf{UGStudent} \sqcup \mathsf{GStudent} \\ & \mathsf{UGStudent} \sqsubseteq \exists \mathsf{major} \\ & \mathsf{GStudent} \sqsubseteq \exists \mathsf{faculty} \\ \mathcal{A} :\ & \mathsf{Student(Mary)} \\ \mathcal{DB} :\ & \underline{\mathsf{major}} \qquad \underline{\mathsf{faculty}}\end{aligned} \tag{15}$$

then the KB is still inconsistent. Mary does not have to be either known to be an undergraduate or known to be a graduate, but because of the disjunctive definition of Student she does have to be either an undergraduate, in which case she has no major, or a graduate, in which case she has no faculty. Both cases lead to an inconsistency.

4 RDF(S) and DBoxes

Our development of constraints using DBoxes and completely specified concepts and roles has used OWL (or $\mathcal{SROIQ}(\mathcal{D})$) as the ontology language. We used OWL for two reasons. First, previous work on constraints in ontology languages has concentrated on OWL or other expressive ontology languages, so using OWL here allows us to make better comparisons with previous work. Second, DBoxes can be rewritten as other OWL constructs, showing that DBoxes do add not any expressive power to OWL.

It is possible to use RDF or RDFS [3] as the ontology language for DBoxes. The basic idea is that for any URI D declared to be a DBox predicate in $\mathcal{S}(\mathcal{DB})$, the set of the URIs C_i stated *explicitly* in the graph as having D as their `rdf:type` – namely the set of all C_i appearing in triples of the form (C_i `rdf:type` D) – has to be considered as the *complete* set of instances of D. DBoxes extend the expressive power of RDF and RDFS because in DBoxes is it possible to infer that facts are false, e.g., any non-stated fact for a DBox concept or role. This addition of expressive power makes reasoning in RDF(S) plus DBoxes harder than reasoning in just RDF(S); as a matter of fact, we can prove the following theorem.

Theorem (Complexity of RDF(S) with DBoxes). *SPARQL query answering with basic graph patterns (BGPs) under the RDF simple entailment regime, and the RDFS entailment regime,* augmented with DBoxes *is coNP-hard for data complexity.*

The proof is based on a reduction to the 3-colorability problem, a reduction similar to the one employed in [7].

The use of DBoxes permits RDFS to represent the full meaning of database information that is imported into RDFS, adding an important aspect to RDFS. Even with the expressive weakness of RDFS, it is possible to make the kind of inferences that have been argued to be problematic, for example by using role domains or ranges to infer concept membership. The use of DBoxes turn role domain and range axioms into constraints, allowing the elimination of these inferences in cases where they might be problematic.

5 Conclusion

There is no perfect approach to integrity constraints in an ontology setting. One would like to have the situation in databases, where the behavior of integrity constraints is precisely a check against data, while still retaining the open nature of ontology languages. This is not possible because the open nature of ontology languages means that axioms make inferences and do not simply check the data. One would also like to be able to check integrity constraints quickly, but this is also not possible. In an open setting checking an integrity constraint is at least as costly as determining the consistency of the KB.

Previous approaches to integrity constraints involving autoepistemic constructs use the autoepistemic constructs to determine what is known (in effect, closing off the current knowledge) and utilize this closed version to ensure that the integrity constraints do not add new information. However, autoepistemic approaches make the constraints operate on too few individuals, e.g., only on known instances of a concept, which limits their ability to truly check that the constraints hold.

Approaches involving minimal models have different problems. Instead of having the constraints active on too few individuals, the constraints are too easy to satisfy. Without the use of a special predicate, existential, or unknown, fillers are acceptable in constraints. Even with the solution to the previous problem, disjunctive information is adequate to satisfy a constraint. Further, the minimization needed in the approach is sensitive to the presence of extra concepts in the KB, and axioms that should be irrelevant can change whether a constraint is satisfied or not.

Approaches that involve both minimization (for example of just equality) and modal notions fall prey to versions of the two different approaches. Modal evaluation of constraints means that the constraints often are active on too few individuals. Even minimization of just equality is sensitive to irrelevant axioms.

Our approach to constraints is to completely specify certain concepts and roles, making them into the analogue of database tables. On these concepts and roles, axioms act just like integrity constraints. Although this approach may appear to be without computation cost, the extensive use of nominals in the translation of DBoxes to regular ontology languages can easily stress current ontology reasoners. As well, because DBoxes are closed, adding new information to a DBox concept or role is a modification of the KB, not just an addition.

We hope that future work on ontology reasoners will provide optimizations for both extensive use of nominals and modifications to the KB. Both of these situations occur commonly, and are not specific to DBoxes.

On the other hand, querying DBox concepts and roles is easy, as it is just database querying. Further, answers returned by such queries are complete (as opposed to the situation with other concepts and roles, where the information returned might not be complete) and can be used in applications just like answers to database queries can.

Even with the computational issue, we believe that the DBox approach is the correct approach to providing integrity constraints for ontology languages. The DBox approach provides precisely the same effect for integrity constraints as is the case in databases, which provide the model for integrity constraints. It thus avoids the problems with the other approaches, and thus appears to us to be preferable.

References

1. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, implementation, and applications, 2nd edn. Cambridge University Press (2010)
2. Beth, E.W.: On Padoa's methods in the theory of definitions. Indagotiones Mathematicae 15, 330–339 (1953)
3. Brinkley, D., Guha, R.V.: RDF vocabulary description langauge 1.0; RDF schema. W3C Recommendation (February 2004), `http://www.w3.org/TR/rdf-schema/`
4. de Bruijn, J., Polleres, A., Lara, R., Fensel, D.: OWL Flight. Deliverable D20.3v0.1, WSML (August 2004), `http://www.wsmo.org/TR/d20/d20.3/v0.1`
5. de Bruijn, J., Polleres, A., Lara, R., Fensel, D.: OWL DL vs. OWL Flight: Conceptual modeling and reasoning on the semantic web. In: Proceedings of the Fourteenth International World Wide Web Conference (WWW 2005), pp. 623–632 (May 2005)
6. Donini, F.M., Nardi, D., Rosati, R.: Description logics of minimal knowledge and negation as failure. ACM Transactions on Computational Logic 3(2), 177–225 (2002)
7. Franconi, E., Ibáñez-García, Y.A., Seylan, I.: Query answering with DBoxes is hard. Electronic Notes on Theoretical Computer Science 278, 71–84 (2011)
8. Franconi, E., Kerhet, V., Ngo, N.: Exact Query Reformulation with First-Order Ontologies and Databases. In: del Cerro, L.F., Herzig, A., Mengin, J. (eds.) JELIA 2012. LNCS, vol. 7519, pp. 202–214. Springer, Heidelberg (2012)
9. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible SROIQ. In: Doherty, P., Mylopoulos, J., Welty, C.A. (eds.) Proceedings, Tenth International Conference on Principles of Knowledge Representation and Reasoning, pp. 57–67. AAAI Press (June 2006)
10. Motik, B., Horrocks, I., Sattler, U.: Bridging the gap between OWL and relational databases. Journal of Web Semantics 7(2), 74–119 (2009)
11. Motik, B., Patel-Schneider, P.F., Parsia, B.: OWL 2 web ontology language: Structural specification and functional-style syntax. W3C recommendation (October 2009), `http://www.w3.org/TR/owl2-syntax`
12. Motik, B., Rosati, R.: Reconciling description logics and rules. Journal of the ACM 57(5), 1–62 (2010)
13. Sengupta, K., Krisnadhi, A.A., Hitzler, P.: Local Closed World Semantics: Grounded Circumscription for OWL. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part I. LNCS, vol. 7031, pp. 617–632. Springer, Heidelberg (2011)
14. Seylan, I., Franconi, E., de Bruijn, J.: Effective query rewriting with ontologies over DBoxes. In: Boutilier, C. (ed.) Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI 2009), pp. 923–929 (July 2009)
15. Tao, J., Sirin, E., Bao, J., McGuinness, D.L.: Integrity constraints in OWL. In: Proceedings of the Twenty-Fourth National Conference on Artificial Intelligence, Atlanta, Georgia. American Association for Artificial Intelligence (July 2010)