On Order Dependencies for the Semantic Web

David Toman[†‡] and Grant Weddell[†]

[†]David R. Cheriton School of Computer Science
University of Waterloo, Canada
Email: {david,gweddell}@uwaterloo.ca

[‡]Faculty of Computer Science
Free University of Bozen-Bolzano, Italy
Email: david@inf.unibz.it

Abstract. We consider the problem of adding both equality and order generating dependencies to Web ontology languages such as OWL DL that are based on description logics. Such dependencies underlie a number of problems that relate, for example, to web service composition, to document ordering, and to lower level algorithmic issues in service plan generation and evaluation.

1 Introduction

RDF underlies a vision of the Semantic Web in which both data and metadata are viewed as a set of *subject/property/object* triples that can be associated with web resources denoted by *Universal Resource Identifiers* (URIs) [13]. To support reasoning, there has been a progression of further standards for inferring the existence of additional triples. This is accomplished by adding interpretations for particular RDF properties. For example, in the case of property *subClassOf*, the RDF Schema standard mandates inferring the triple

$$x/\mathit{subClassOf}/\text{ITEM}$$

from the pair of triples

$$x/\mathit{subClassOf}/\text{SALEITEM}$$

and

$$\text{SALEITEM}/\mathit{subClassOf}/\text{ITEM}, \tag{1}$$

where ITEM and SALEITEM are now viewed as *concepts*.

The current best practices for these standards, measured in terms of established reasoning technology, are the *description logic* (DL) based fragments of the OWL web ontology language, called OWL Lite and OWL DL [22]. Building on RDF Schema, they enable a collection of triples to encode more general concepts such as *anything not on sale* or *an item with a reliable supplier*. In this

paper, we use a more abstract and compact syntax developed for description logics in which collections of triples encoding such concepts can be specified more succinctly as

$$\neg\text{SALEITEM}$$

and as

$$\text{ITEM} \sqcap \forall \mathit{Supplier}.\text{RELIABLE},$$

respectively. We do this also for *subClassOf* RDF triples, such as (1) above, which we now refer to as *inclusion dependencies*, and write instead as

$$\text{CONCEPT}_1 \sqsubseteq \text{CONCEPT}_2. \tag{2}$$

For example, (1) above is now written SALEITEM $\sqsubseteq$ ITEM.

A collection of inclusion dependencies of the form (2) defines an OWL DL *ontology* that can be used by other protocols for the semantic web, such as the RDF query language SPARQL [15]. In this setting, XML can be used as a transport language for RDF in the sense that an XML document is an ordered forest of RDF triples that in turn encode OWL DL concepts and inclusion dependencies. However, the significance of *ordering* in XML is currently beyond any real capacity of OWL DL, or even full OWL, to account for any consequent logical significance, e.g., to inform a SPARQL query engine by way of an ontology that the order in which various ITEM concepts occur in a document correlates in some way with their *Price*.

An obvious approach to remedy this is to reconsider the underlying description logic for OWL DL, to consider in particular how concept descriptions can be enriched to capture metadata relating to order. Knowledge of the relevance of document order in XML would not only be useful to a SPARQL query engine in helping to address lower level algorithmic and performance issues, e.g., avoiding sort costs when reporting on the supplier for a given item in order of increasing cost, but also by other web services that reason in turn about web service composition in which attributes are used to abstract temporal artifacts such as events.

In this paper, we consider a new concept constructor for description logics with the potential of endowing OWL DL with an ability to capture knowledge about ordering. Instances of this constructor are called *path order dependencies* (PODs). They are a generalization of *path functional dependencies* (PFDs) that have been considered in the context of a DL dialect called $\mathcal{DLF}$ [19–21], which we also use. This dialect is *feature based* and therefore more functional in style as opposed to the more common *role based* derivatives of $\mathcal{ALC}$ such as OWL DL. As a consequence, it is much easier to incorporate PODs.

Example 1 To illustrate using PODs, consider an ontology of ITEM concepts of relevance, say, to an online supplier of photography equipment. The supplier maintains an XML document of the ITEM concepts in such a way that subtrees defining the concepts satisfy a major sort on their *ProductCode* feature

and a minor sort on their *Price* feature. This knowledge can now be captured by an inclusion dependency using two instances of the proposed POD concept constructor as follows:

$$\begin{aligned} \text{ITEM} \quad \sqsubseteq \quad & (\text{ITEM} : \{DocOrder^{<}\} \rightarrow \{ProductCode^{\leq}\}) \\ & \sqcap \ (\text{ITEM} : \{DocOrder^{<}, ProductCode^{=}\} \rightarrow \{Price^{\leq}\}). \end{aligned}$$

As a second example, the supplier in question can capture an inherent ordering for SALEITEM concepts in which their relative ordering by virtue of their *Price* is preserved by their *DiscountPrice* by adding the following:

$$\begin{aligned} \text{SALEITEM} \quad \sqsubseteq \quad & \text{ITEM} \\ & \sqcap \ (\text{SALEITEM} : \{Price^{<}\} \rightarrow \{DiscountPrice^{<}\}. \end{aligned}$$

Note that, in comparison to OWL DL, $\mathcal{DLF}$ is a worthwhile basis for study since it is already sufficient to simulate the DL dialect $\mathcal{ALCQI}$ in an intuitive fashion using *role reification* [19]. In essentially the same way, $\mathcal{ALCQI}$ can in turn simulate $\mathcal{SHIQ}$ without transitive roles, a large subset of OWL DL that includes OWL Lite. With regard to the above hypothetical metadata about item ordering in a document, the examples are expressed in terms of the DL dialect $\mathcal{DLFD}_{reg}$, the extension of $\mathcal{DLF}$ considered in this paper.

Our contributions relate to $\mathcal{DLFD}_{reg}$ and are as follows.

1. We define a guarded condition for PODs for which the associated implication problem remains decidable and indeed unchanged from $\mathcal{DLF}$; and
2. We show how a slight relaxation of this condition leads to undecidability.

1.1 Related Work

In addition to OWL DL, description logics have been used extensively as a formal way of understanding a large variety of languages for specifying meta-data, including ER diagrams, UML class and object diagrams, relational database schemata, and so on [14].

The form of order dependencies introduced in this paper is a generalization of a relational variant [17], and is also a generalization of regular PFDs introduced in [19]. Less expressive first order PFDs were introduced and studied in the context of object-oriented data models [8, 23]. An FD concept constructor was proposed and incorporated already in Classic [4], an early DL with a PTIME reasoning procedure, without changing the complexity of its implication problem. The generalization of this constructor to PFDs alone leads to EXPTIME completeness of the implication problem [10]; this complexity remains unchanged in the presence of additional concept constructors common in rich DLs such as roles, qualified number restrictions, and so on [17, 18].

In [5], the authors consider a DL with functional dependencies and a general form of keys added as additional varieties of dependencies, called a *key box*. They

show that their dialect is undecidable for DLs with inverse roles, but becomes decidable when unary functional dependencies are disallowed. This line of investigation is continued in the context of PFDs and inverse features, with analogous results [20]. We therefore disallow inverse features in this paper to exclude an already known cause for undecidability.

PFDs have also been used in a number of applications in object-oriented schema diagnosis and synthesis [2, 3], in query optimization [6, 9] and in the selection of indexing for a database [16].

Order dependencies have been considered in the context of the relational model [7], and as a special case of constraint-generating dependencies for the relational model [1]. A form of key dependency with left hand side feature paths has been considered for a DL coupled with various concrete domains [12, 11]. In this case, the authors explore how the complexity of satisfaction is influenced by the selection of a particular concrete domain together with various syntactic restrictions on the key dependencies themselves. Note that this earlier work strictly separates objects that serve as "domain values," and can therefore be ordered, from abstract objects such as tuples. This makes such approaches less applicable in the RDF setting in which no such distinction exists, where both objects and values can in turn be object-attribute-value triples.

The remainder of the paper is organized as follows. Section 2 that follows defines the syntax and semantics for $\mathcal{DLFD}_{\text{reg}}$. Our main results are then presented in Section 3. We conclude with a summary and a discussion of remaining issues and open problems in Section 4.

2 Definitions

The syntax and semantics of the $\mathcal{DLFD}_{\text{reg}}$ dialect of description (or feature) logics are given by the following.

Definition 2 (Syntax and Semantics of $\mathcal{DLFD}_{\text{reg}}$) Let F be an arbitrary finite set of attribute names. We define a *path language* L to be a regular language over the alphabet F. We use *regular expressions* as the surface syntax for such languages with Id standing for the empty word in L. We use $L^{\approx}$ to denote a regular language in which every word $\mathsf{Pf}^{\sim} \in L^{\approx}$ is a concatenation of a word from $\mathsf{Pf} \in L$ with a symbol $\sim \in \{<, \leq, =, \geq, >\}$. We denote by $L^{\sim}$ the regular sublanguage $\{\mathsf{Pf}^{\sim} \mid \mathsf{Pf}^{\sim} \in L^{\approx}\}$ in which all words end with the same symbol $\sim$.

Let C be primitive concept description(s). We define *derived concept descriptions* using the grammar in Figure 2. A concept formed by an application of the final production in the grammar is called a *regular path order dependency* (POD).

An *inclusion dependency* $\mathcal{C}$ is an expression of the form $D \sqsubseteq E$.

The *semantics* of expressions is given with respect to a structure $(\Delta, \leq, \cdot^{\mathcal{I}})$, where Δ is a domain of "objects"; $\leq$ is a linear order on Δ; and $(.)^{\mathcal{I}}$ an interpretation

Syntax:			Semantics: Defn of "$(\cdot)^{\mathcal{I}}$"
D	::=	C	$(C)^{\mathcal{I}} \subset \Delta$
	|	$D_1 \sqcap D_2$	$(D_1)^{\mathcal{I}} \cap (D_2)^{\mathcal{I}}$
	|	$\forall L.D$	$\bigcap_{\mathsf{Pf} \in L} \{x : (\mathsf{Pf})^{\mathcal{I}}(x) \in (D)^{\mathcal{I}}\}$
	|	$\neg D$	$\Delta \setminus (D)^{\mathcal{I}}$
E	::=	D	
	|	$E_1 \sqcap E_2$	$(E_1)^{\mathcal{I}} \cap (E_2)^{\mathcal{I}}$
	|	$E_1 \sqcup E_2$	$(E_1)^{\mathcal{I}} \cup (E_2)^{\mathcal{I}}$
	|	$\forall L.E$	$\bigcap_{\mathsf{Pf} \in L} \{x : (\mathsf{Pf})^{\mathcal{I}}(x) \in (E)^{\mathcal{I}}\}$
	|	$D : L_1^{\approx} \rightarrow L_2^{\approx}$	$\{x : \forall\, y \in (D)^{\mathcal{I}}. \bigwedge_{\mathsf{Pf}^{\sim} \in L_1^{\approx}} (\mathsf{Pf})^{\mathcal{I}}(x) \sim (\mathsf{Pf})^{\mathcal{I}}(y) \Rightarrow \bigwedge_{\mathsf{Pf}^{\sim} \in L_2^{\approx}} (\mathsf{Pf})^{\mathcal{I}}(x) \sim (\mathsf{Pf})^{\mathcal{I}}(y)\}$

Fig. 1. Syntax and Semantics of $\mathcal{DLFD}_{\mathrm{reg}}$.

function that fixes the interpretations of primitive concepts to be subsets of Δ and of primitive attributes in F to be total functions over Δ. The interpretation is extended to words over F as follows: $(Id)^{\mathcal{I}} = \lambda x.x$ and $(f.\,\mathsf{Pf})^{\mathcal{I}} = (\mathsf{Pf})^{\mathcal{I}} \circ (f)^{\mathcal{I}}$, and to derived concept descriptions, cf. Figure 2.

An interpretation *satisfies an inclusion dependency* $\mathcal{C}$ *of the form* $D \sqsubseteq E$ if $(D)^{\mathcal{I}} \subseteq (E)^{\mathcal{I}}$.

A *terminology* $\mathcal{T}$ consists of a finite set of inclusion dependencies. The *logical implication problem* asks if $\mathcal{T} \models \mathcal{C}$ holds; that is, if all interpretations that satisfy each constraint in $\mathcal{T}$ must also satisfy $\mathcal{C}$ (the posed question).

Note that the notation $\mathsf{Pf}^{\sim} \in L^{\approx}$ stands for the fact that *the path (string)* $\mathsf{Pf}^{\sim}$ *belongs to the language* $L^{\approx}$. The paths are in turn interpreted as (compositions of) total functions over the domain Δ. Hence the conjunctions in the semantic definition of a POD range over all words in an appropriate regular language and define order among objects in the range of their interpretations.

The two-level syntax is needed to prevent any occurrence of a POD on the left-hand side of an inclusion dependency or within the scope of negation. Removing this restriction leads to undecidability [21].

Example 3 Recall our introductory example relating to ITEM concepts maintained by a hypothetical online supplier. Now suppose the supplier has a second XML document containing a sequence of subtrees encoding SUPPLIES concepts, and that this document satisfies the following property:

> *a traversal of the root nodes for the* SUPPLIES *elements correlates with a major sort of the* ITEM *component of each element, and a minor sort of the wholesale price.*

When added to a terminology, the following inclusion dependency formally captures this property:

$$\begin{aligned} \text{SUPPLIES} \;\sqsubseteq\; & \forall \mathit{Iref}.\text{ITEM} \\ & \sqcap\; (\text{SUPPLIES} : \{\mathit{DocOrder}^{<}\} \rightarrow \{\mathit{Iref}^{\leq}\}) \\ & \sqcap\; (\text{SUPPLIES} : \{\mathit{DocOrder}^{<}, \mathit{Iref}^{=}\} \rightarrow \mathit{WholesalePrice}^{\leq}). \end{aligned}$$

To paraphrase the final line: *if the first of a pair of arbitrary* SUPPLIES *concepts precedes the second in a given document and if both refer to the same items, then the wholesale price of the first will not exceed the wholesale price of the second.*

3 Reasoning in $\mathcal{DLFD}_{\text{reg}}$

The question of *logical implication* is central to the use of logic-based approaches to conceptual modeling of the artifacts in the semantic Web. This section shows the main results relating to the logical implication problem with respect to PODs.

3.1 Undecidability for General Order Dependencies

The general implication problem for $\mathcal{DLFD}_{\text{reg}}$ is, unfortunately, undecidable:

Proposition 4 ([21]) *The implication problem for $\mathcal{DLFD}_{\text{reg}}$ becomes undecidable when dependencies of the form $D : \{\} \rightarrow \{f^{=}\}$ are allowed. This is the case even when all dependencies are restricted to finite languages and are equality-generating.*

Path-functional dependencies with empty left-hand sides allow one to simulate *nominals*—concept descriptions whose interpretation must correspond to a singleton set; this can be enforced, e.g., for a concept C, by the inclusion dependency $C \sqsubseteq C : \{\} \rightarrow \{\mathit{Id}^{=}\}$.

Decidability can be reobtained by requiring any regular languages occurring in PFDs to be non-empty [19]. However, this restriction does not suffice for the more general case of PODs.

Theorem 5 *The implication problem for $\mathcal{DLFD}_{\text{reg}}$ is undecidable. This remain true when all regular languages occurring in PODs are non-empty.*

Proof: (sketch) The above dependency with an empty left-hand side can be simulated by the order dependency $C : \mathit{Id}^{\leq} \rightarrow \{\mathit{Id}^{=}\}$. The remainder follows from a reduction of a tiling problem to the implication problem, expanding on the reduction proposed in [21]. □

3.2 Decidability for Guarded Order Dependencies

To regain decidability, we define a subset of PODs called *guarded PODs*. Intuitively, we require all the PODs appearing in the terminology to be satisfied by trees whose nodes are ordered by the $\leq$ relation top-down and left-to-right (breadth-first).

Definition 6 (Guarded Order Dependency)
An order dependency $D_1 \sqsubseteq D_2 : L_1^{\approx} \rightarrow L_2^{\approx}$ *is* guarded *if it satisfies the following conditions:*

1. *if* $= \subseteq \sim$ *for all* $\mathsf{Pf}^{\sim} \in L_1^{\approx}$ *then also* $= \subseteq \sim$ *for all* $\mathsf{Pf}^{\sim} \in L_2^{\approx}$,
2. *if* $< \subseteq \sim$ *for all* $\mathsf{Pf}^{\sim} \in L_1^{\approx}$ *then also* $< \subseteq \sim$ *for all* $\mathsf{Pf}^{\sim} \in L_2^{\approx}$, *and*
3. *if* $> \subseteq \sim$ *for all* $\mathsf{Pf}^{\sim} \in L_1^{\approx}$ *then also* $> \subseteq \sim$ *for all* $\mathsf{Pf}^{\sim} \in L_2^{\approx}$,

where $\subseteq$ *denotes set inclusion among the interpretations of the binary relations denoted by* $\{<, \leq, =, \geq, >\}$.

For the remainder of the paper, we assume all PODs are guarded. The ramification of definition is that guarded order dependencies in a terminology cannot, on their own, lead to inconsistency. This is in contrast to the general case where, e.g., $\top \sqsubseteq \top : \{f^{<}\} \rightarrow \{f^{>}\}$ is not satisfiable.

To aid the decision procedure for the guarded case, we simplify terminologies of $\mathcal{DLFD}_{\text{reg}}$ implication problems as follows:

Definition 7 *A* $\mathcal{DLFD}_{\text{reg}}$ *implication problem* $\mathcal{T} \models \mathcal{C}$ *is* simple *if each inclusion dependency in* $\mathcal{T}$ *is of the form* $D_1 \sqsubseteq D_2$ *or the form* $D_1 \sqsubseteq D_2 : L_1^{\approx} \rightarrow L_2^{\approx}$. *In the former case, the dependency is described as* pure*; in the latter case, the dependency is called an* order dependency*.*

It is easy to see that unrestricted implication problems can be always reduced to reasoning w.r.t. simple terminologies only—called *simple* implication problems:

Lemma 8 *Let* $\mathcal{T}$ *be an arbitrary* $\mathcal{DLFD}_{\text{reg}}$ *terminology and* $\mathcal{C}$ *an arbitrary subsumption constraint. Then there is a simple terminology* $\mathcal{T}'$ *such that* $\mathcal{T} \models \mathcal{C}$ *if and only if* $\mathcal{T}' \models \mathcal{C}$.

Proof: (sketch) $\mathcal{T}'$ introduces additional primitive concept descriptions to *name* subconcepts on the right-hand sides of concept descriptions in $\mathcal{T}$. □

For each simple $\mathcal{DLFD}_{\text{reg}}$ implication problem $\mathcal{T} \models \mathcal{C}$, we define a corresponding $\mathcal{DLF}_{\text{reg}}$ satisfiability problem. There are two cases to consider depending on $\mathcal{C}$.

Pure Posed Questions. For simple terminologies that use guarded ordered dependencies only and for a pure constraint $\mathcal{C}$, the logical implication problem can be reduced to the implication problem that does not involve ordered dependencies:

Lemma 9 *Let $\mathcal{T}$ be a simple $\mathcal{DLFD}_{\text{reg}}$ terminology and $\mathcal{C}$ a pure inclusion dependency. Also let $\mathcal{T}'$ be the set of all pure inclusion dependencies in $\mathcal{T}$. Then $\mathcal{T} \models \mathcal{C}$ if and only if $\mathcal{T}' \models \mathcal{C}$.*

Proof: Consider a tree model of $\mathcal{T}' \cup \{\neg\mathcal{C}\}$ with nodes ordered by their breadth-first traversal number. This model satisfies all (possible) guarded order dependencies, hence it is a model of $\mathcal{T} \cup \{\neg\mathcal{C}\}$. The other direction is immediate as $\mathcal{T}' \subseteq \mathcal{T}$. □

The decidability of this problem is then an immediate consequence of the following proposition, since $\mathcal{T}'$ is a $\mathcal{DLF}_{\text{reg}}$ terminology.

Proposition 10 ([19]) *The implication problem for $\mathcal{DLF}_{\text{reg}}$ is decidable and complete for EXPTIME.*

In addition, whenever $\mathcal{T} \not\models D \sqsubseteq D'$, there is a F-tree with nodes labeled by sets of concept descriptions that serves as a model of $\mathcal{T}$ and whose root label contains the concepts D and $\neg D'$.

General Posed Questions. Due to the undecidability issues connected with allowing order dependencies in the scope of negation, it is not possible to express a negation of a posed question as a concept description. We develop an alternative solution, based on construction in [8, 24]. We introduce the solution by an example.

Example 11 Consider a terminology $\mathcal{T}$ and a posed question of the form $D \sqsubseteq D' : L_1^{\approx} \rightarrow L_2^{\approx}$. To falsify such an order dependency, *two* objects are needed, one in the interpretation of D and another in the interpretation of D'. Hence, by Lemma 9, both D and D' must be satisfiable with respect to $\mathcal{T}'$, the *pure part* of $\mathcal{T}$. Note that the two models witnessing the satisfiability of D and D', if they exist, are F-trees that differ only in the labeling of nodes by concept descriptions.

To simulate the two models and the effects of the posed question using only a single F-tree, we define a $\mathcal{DLF}_{\text{reg}}$ terminology consisting of the following components that simulate the effects of the original assertions in this new interpretation:

- $\mathcal{T}_1'$ and $\mathcal{T}_2'$, that are two copies of $\mathcal{T}'$ in which all primitive concept descriptions C have been renamed to C_1 and C_2, respectively;

– $\mathcal{T}_{1,2}$, that captures the effects of order dependencies in $\mathcal{T}$ on the two interpretations. These effects are captured by auxiliary primitive concept descriptions $\mathsf{Aux}^{\sim}_{1,2}$ and $\mathcal{DLF}_{\text{reg}}$ constraints of the form

$$((D_1 \sqcap D_2') \sqcup (D_1' \sqcap D_2)) \sqcap (\forall L_1^{<}.\mathsf{Aux}^{<}_{1,2}) \sqcap \ldots \sqcap (\forall L_1^{>}.\mathsf{Aux}^{>}_{1,2}) \sqsubseteq (\forall L_2^{<}.\mathsf{Aux}^{<}_{1,2}) \sqcap \ldots \sqcap (\forall L_2^{>}.\mathsf{Aux}^{>}_{1,2})$$

created for each $D \sqsubseteq D' : L_1^{\approx} \rightarrow L_2^{\approx} \in \mathcal{T}$ where $L_i^{\approx} = L_i^{<} \cup \ldots \cup L_i^{>}$ is a partition of $L_i^{\approx}$ according to the order predicate associated with the individual words (and for $i = 1, 2$). Intuitively, membership in $\mathsf{Aux}^{\sim}_{1,2}$ concepts stands for the $\sim$ relationship between *corresponding* objects in the two tree interpretations that are encoded by this model.

– A terminology $\mathcal{A}$ of auxiliary assertions that govern the interactions of the $\mathsf{Aux}^{\sim}_{1,2}$ concepts in accordance with the axioms of linear order. In addition, assertions governing the *existence* of nodes in the *copies* of the tree are also included here (e.g., the fact that in an actual counterexample, such as the one in Figure 2, the rightmost root node is not necessarily present).

Counterexamples to the posed question $D \sqsubseteq D' : L_1^{\approx} \rightarrow L_2^{\approx}$ are then captured as objects satisfying the concept

$$D_1 \sqcap D_2' \sqcap (\forall L_1^{<}.\mathsf{Aux}^{<}_{1,2}) \sqcap \ldots \sqcap (\forall L_1^{>}.\mathsf{Aux}^{>}_{1,2}) \sqcap \neg((\forall L_2^{<}.\mathsf{Aux}^{<}_{1,2}) \sqcap \ldots \sqcap (\forall L_2^{>}.\mathsf{Aux}^{>}_{1,2})).$$

Hence the logical implication is reduced to concept satisfiability w.r.t. a modified terminology.

However, allowing arbitrary $\mathcal{DLFD}_{\text{reg}}$ inclusion dependencies as posed questions, e.g., in which order dependencies occur within other positive concept constructors, involves an additional construction which extends an earlier form used in the simpler case of path-functional dependencies [21]:

Example 12 A counterexample to the constraint

$$A \sqsubseteq (B : \{f f^{<}\} \rightarrow \{f g^{<}\}) \sqcup \forall\{f\}.(C : \{f^{>}\} \rightarrow \{g^{>}\})$$

is shown in Figure 2. Observe with this case that the distinct C object must occur at *different* levels when compared to an A-rooted forest. Such a counterexample, however, cannot be constructed in the presence of a terminology $\{B \sqsubseteq \forall\{f\}.C, C \sqsubseteq C : \{f^{>}\} \rightarrow \{g^{>}\}\}$. Hence the example posed question is a logical consequence of this terminology.

The examples suggest a need for multiple *root objects* in counterexample interpretations, with the roots themselves occurring at different levels. Our overall strategy is to therefore reduce a logical implication problem to a negation of a consistency problem in an alternative formulation in which objects in a satisfying counterexample *denote up to ℓ possible copies in a counterexample interpretation*

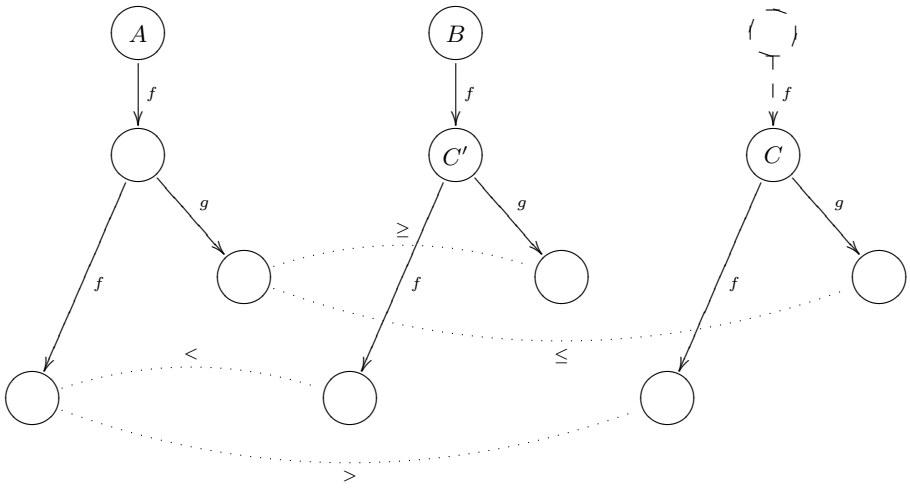

Fig. 2. Counterexample for Example 12.

for the original problem, where ℓ is the number of occurrences of PODs in the posed question.

To encode this one-to-many mapping of objects, we require a general way to have ℓ copies of concepts occurring in a given membership problem. We therefore write D_i to denote the concept description D in which all primitive concepts C are replaced by C_i. For a simple terminology $\mathcal{T}$ we then define

$$\mathcal{T}^i = \{\mathsf{Nd}_i \sqcap D_i \sqsubseteq E_i \mid D \sqsubseteq E \in \mathcal{T} \text{ pure}\}, \text{ and}$$

$$\begin{aligned}\mathcal{T}^{i,j} = \{&\mathsf{Nd}_i \sqcap \mathsf{Nd}_j \sqcap D_i \sqcap D'_j \sqcap \Big(\bigsqcap_{L_1^{\sim} \subseteq L_1^{\approx}} \forall L_1^{\sim}.\mathsf{Aux}_{0,i}^{\sim}\Big) \sqsubseteq \Big(\bigsqcap_{L_2^{\sim} \subseteq L_2^{\approx}} \forall L_2^{\sim}.\mathsf{Aux}_{0,i}^{\sim}\Big),\\ &\mathsf{Nd}_i \sqcap \mathsf{Nd}_j \sqcap D_j \sqcap D'_i \sqcap \Big(\bigsqcap_{L_1^{\sim} \subseteq L_1^{\approx}} \forall L_1^{\sim}.\mathsf{Aux}_{0,i}^{\sim}\Big) \sqsubseteq \Big(\bigsqcap_{L_2^{\sim} \subseteq L_2^{\approx}} \forall L_2^{\sim}.\mathsf{Aux}_{0,i}^{\sim}\Big)\\ &\qquad\qquad \mid D \sqsubseteq D' : L_1^{\approx} \to L_2^{\approx} \in \mathcal{T}\ \}.\end{aligned}$$

For a concept description E we define

$$\mathsf{ToCon}(E) = \begin{cases} D_0 & \text{if } E\ (= D) \text{ is POD free,}\\ \mathsf{ToCon}(E_1) \sqcap \mathsf{ToCon}(E_2) & \text{if } E = E_1 \sqcap E_2,\\ \mathsf{ToCon}(E_1) \sqcup \mathsf{ToCon}(E_2) & \text{if } E = E_1 \sqcup E_2,\\ \forall L.\mathsf{ToCon}(E_1) & \text{if } E = \forall L.E_1, \text{ and}\\ \neg(\mathsf{Nd}_i \sqcap D_i \sqcap \Big(\bigsqcap_{L_1^{\sim} \subseteq L_1^{\approx}} \forall L_1^{\sim}.\mathsf{Aux}_{0,i}^{\sim}\Big)) \sqcup \Big(\bigsqcap_{L_2^{\sim} \subseteq L_2^{\approx}} \forall L_2^{\sim}.\mathsf{Aux}_{0,i}^{\sim}\Big) & \\ \qquad\qquad \text{otherwise, when } E = D : L_1^{\approx} \to L_2^{\approx} \end{cases}$$

where i in the last equation is the index of the POD in the original posed question.

In the above, we have introduced primitive concepts $\mathsf{Aux}^{\sim}_{i,j}, 0 \leq i \neq j \leq \ell$, to express that the ith and jth object copies are related by $\sim$, and $\mathsf{Nd}_i, 0 \leq i \leq \ell$, to assert that the ith copy exists. The following auxiliary sets of constraints are therefore defined to account for the axioms of equality and of linear orders, and for the fact that features in $\mathcal{DLFD}_{\mathrm{reg}}$ denote total functions.

$$
\begin{aligned}
\mathcal{A}(\ell) \;=\; & \{\mathsf{Aux}^{\sim}_{i,j} \sqcap \mathsf{Aux}^{\sim}_{j,k} \sqsubseteq \mathsf{Aux}^{\sim}_{i,k} \mid 0 \leq i < j < k \leq \ell\,\} \\
\cup\; & \{\mathsf{Aux}^{<}_{i,j} \sqcap \mathsf{Aux}^{=}_{i,j} \sqsubseteq \bot, \mathsf{Aux}^{<}_{i,j} \sqcap \mathsf{Aux}^{>}_{i,j} \sqsubseteq \bot, \\
& \qquad\qquad \mathsf{Aux}^{=}_{i,j} \sqcap \mathsf{Aux}^{>}_{i,j} \sqsubseteq \bot \mid 0 \leq i \neq j \leq \ell\,\} \\
\cup\; & \{\top \sqsubseteq \mathsf{Aux}^{<}_{i,j} \sqcup \mathsf{Aux}^{=}_{i,j} \sqcup \mathsf{Aux}^{>}_{i,j} \mid 0 \leq i \neq j \leq \ell\,\} \\
\cup\; & \{\mathsf{Aux}^{=}_{i,j} \sqsubseteq \mathsf{Aux}^{=}_{j,i}, \mathsf{Aux}^{<}_{i,j} \sqsubseteq \mathsf{Aux}^{>}_{j,i} \mid 0 \leq i \neq j \leq \ell\,\} \\
\cup\; & \{\mathsf{Aux}^{\leq}_{i,j} \sqsubseteq \mathsf{Aux}^{<}_{j,i} \sqcup \mathsf{Aux}^{=}_{i,j}, \\
& \qquad\qquad \mathsf{Aux}^{=}_{i,j} \sqsubseteq \mathsf{Aux}^{\leq}_{i,j}, \mathsf{Aux}^{<}_{i,j} \sqsubseteq \mathsf{Aux}^{\leq}_{i,j} \mid 0 \leq i \neq j \leq \ell\,\} \\
\cup\; & \{\mathsf{Aux}^{\geq}_{i,j} \sqsubseteq \mathsf{Aux}^{>}_{j,i} \sqcup \mathsf{Aux}^{=}_{i,j}, \\
& \qquad\qquad \mathsf{Aux}^{=}_{i,j} \sqsubseteq \mathsf{Aux}^{\geq}_{i,j}, \mathsf{Aux}^{>}_{i,j} \sqsubseteq \mathsf{Aux}^{\geq}_{i,j} \mid 0 \leq i \neq j \leq \ell\,\} \\
\cup\; & \{(\mathsf{Aux}^{=}_{i,j} \sqcap C_i) \sqsubseteq C_j \mid 0 \leq i \neq j \leq \ell \text{ and } C \text{ a primitive concept}\} \\
\cup\; & \{\mathsf{Aux}^{=}_{i,j} \sqsubseteq \forall f.\mathsf{Aux}^{=}_{i,j} \mid 0 \leq i \neq j \leq \ell \text{ and } f \in F \text{ a primitive feature}\} \\
\cup\; & \{\mathsf{Nd}_i \sqsubseteq \forall f.\mathsf{Nd}_i \mid 0 \leq i \leq \ell \text{ and } f \in F \text{ a primitive feature}\}
\end{aligned}
$$

Theorem 13 *Let $\mathcal{T}$ be a simple terminology and $D \sqsubseteq E$ an inclusion dependency containing ℓ occurrences of the POD concept constructor. Then $\mathcal{T} \models D \sqsubseteq E$ if and only if*

$$(\bigcup_{0 \leq i \leq \ell} \mathcal{T}_i) \cup (\bigcup_{0 \leq i < j \leq \ell} \mathcal{T}_{i,j}) \cup \mathcal{A}(\ell) \models (\mathsf{Nd}_0 \sqcap D_0 \sqcap \neg \mathsf{ToCon}(E)) \sqsubseteq \bot.$$

Proof: (sketch) Given an interpretation $\mathcal{I}$ such that $\mathcal{I} \models \mathcal{T}$ and $\mathcal{I} \not\models D \sqsubseteq E$ we construct an interpretation $\mathcal{J}$ as follows. First, in the construction, we use a many-to-one map $\delta : \Delta^{\mathcal{I}} \to \Delta^{\mathcal{J}}$ to associate objects in $\mathcal{I}$ with those in $\mathcal{J}$. The range of δ serves as the domain of the interpretation $\mathcal{J}$. For the counterexample object $o \in (D \sqcap \neg E)^{\mathcal{I}}$ we set $\delta o \in (\mathsf{Nd}_0)^{\mathcal{J}}$. Then, for all $o \in \Delta$ and $0 \leq i \neq j \leq \ell$ we define the map δ and the interpretation $\mathcal{I}$ as follows:

- $\delta o \in (\mathsf{Nd}_i)^{\mathcal{J}} \wedge (f)^{\mathcal{I}}(o) = o' \Rightarrow \delta o' \in (\mathsf{Nd}_i)^{\mathcal{J}} \wedge (f)^{\mathcal{J}}(\delta o) = \delta o'$,
- $\delta o \in (\mathsf{Nd}_i)^{\mathcal{J}} \wedge o \in (D)^{\mathcal{I}} \Rightarrow \delta o \in (D)^{\mathcal{J}}$ for D a POD free concept,
- $\delta o = \delta o' \wedge \delta o \in (\mathsf{Nd}_i)^{\mathcal{J}} \wedge \delta o' \in (\mathsf{Nd}^j)^{\mathcal{J}} \wedge (\mathsf{Pf})^{\mathcal{I}}(o) \sim (\mathsf{Pf})^{\mathcal{I}}(o') \Rightarrow \delta o \in (\mathsf{Aux}^{\sim}_{i,j})^{\mathcal{J}}$, and
- $\delta o \in (\mathsf{Nd}_i)^{\mathcal{J}} \wedge o \notin (D : L_1^{\approx} \to L_2^{\approx})^{\mathcal{I}}$ where $D : L_1^{\approx} \to L_2^{\approx}$ is the i-th POD constructor in E. Thus there must be $o' \in \Delta$ such that $o' \in (D)^{\mathcal{I}}$ and

the pair o, o' falsifies the POD; we set $\delta o = \delta o'$ and $\delta o' \in (\neg(\mathsf{Nd}_i \sqcap D_i \sqcap (\sqcap_{L^\sim \subseteq L_1^{\approx}} \forall L^\sim.\mathsf{Aux}^\sim_{0,i})) \sqcup (\sqcap_{L^\sim \subseteq L_2^{\approx}} \forall L^\sim.\mathsf{Aux}^\sim_{0,i}))^{\mathcal{J}}$.

Note that, due to the syntactic restrictions imposed on the uses of POD constructors, a complement of an POD can be enforced only in the counterexample of the description E. Spurious occurrences of negated PODs in the interpretation $\mathcal{I}$ are therefore ignored as the interpretation itself satisfies all PODs in $\mathcal{T}$.

It is easy to verify that $\delta o \in (\mathsf{Nd}_0 \sqcap D_0 \sqcap \neg\mathsf{ToCon}(E))^{\mathcal{J}}$ for $o \in (D \sqcap \neg E)^{\mathcal{I}}$. By inspecting all inclusion dependencies in $\mathcal{T}$ we have $\mathcal{J} \models \mathcal{T}_i$ as $\mathcal{I} \models \mathcal{T}$. Furthermore, the construction of $\mathcal{J}$ enforces $\mathcal{J} \models \mathcal{A}(\ell)$.

On the other hand, given a tree-shaped interpretation $\mathcal{J}$ of $(\mathsf{Nd}_0 \sqcap D_0 \sqcap \mathsf{ToCon}(E))$ that satisfies all assertions in

$$(\bigcup_{0 \le i \le \ell} \mathcal{T}_i) \cup (\bigcup_{0 \le i < j \le \ell} \mathcal{T}_{i,j}) \cup \mathcal{A}(\ell),$$

we construct an interpretation $\mathcal{I}$ of $\mathcal{T}$ that falsifies $D \sqsubseteq E$ as follows:

- $\Delta^{\mathcal{I}} = \{(o, i) : o \in (\mathsf{Nd}_i)^{\mathcal{J}}, 0 \le i \le \ell$ and $o \notin (\mathsf{Aux}^{=}_{j,i})^{\mathcal{J}}$ for any $0 \le j < i\}$,
- $(f)^{\mathcal{I}}((o, i)) = (o', j)$ whenever $(f)^{\mathcal{J}}(o) = o'$ where j is the smallest integer such that $o \in (\mathsf{Aux}^{=}_{j,i})^{\mathcal{J}}$ if such value exists and i otherwise; and
- $(o, i) \in (D)^{\mathcal{I}}$ whenever $(o, i) \in \Delta^{\mathcal{J}}$ and $o \in (D_i)^{\mathcal{J}}$.

The values $(o, i) \in \Delta^{\mathcal{J}}$ are ordered by the breadth-first number of o in $\mathcal{I}$ and then consistently with the interpretation of the $\mathsf{Aux}^\sim_{i,j}$ descriptions in $\mathcal{I}$.

It is easy to verify that $(o, 0)$ falsifies $D \sqsubseteq E$ whenever o belongs to $(\mathsf{Nd}_0 \sqcap D_0 \sqcap \neg\mathsf{ToCon}(E))$, and such an object must exist by our assumptions. Also, $\mathcal{I} \models \mathcal{T}$, as otherwise by cases analysis we get a contradiction with $\mathcal{J} \models (\bigcup_{0 \le i \le \ell} \mathcal{T}_i) \cup (\bigcup_{0 \le i < j \le \ell} \mathcal{T}_{i,j}) \cup \mathcal{A}(\ell)$. □

Corollary 14 *The implication problem for guarded $\mathcal{DLFD}_{\mathrm{reg}}$ is decidable and EXPTIME-complete.*

Proof: Follows immediately from Proposition 10 and Theorem 13 above. □

4 Summary

In this paper, we have explored the possibility of adding a very general form of equality and order generating dependencies based on regular languages to Web ontology languages deriving from description logics. In particular, we have introduced a description logic dialect called $\mathcal{DLFD}_{\mathrm{reg}}$ that incorporates such dependencies as a new concept constructor, and have explored the computational properties of the associated implication problems.

4.1 Remaining Issues and Open Problems

The negative results that relate to the possibility of admitting nominals to $\mathcal{DLFD}_{\text{reg}}$ is unfortunate indeed [21], since OWL DL requires this ability. An important open problem is to devise other restrictions on the PODs concept constructor or on occurrences of this constructor in an implication problem that allows effective reasoning in the presence of nominals.

Another direction of research leads towards tractable dialects. Again, preliminary investigations suggest that there might be a polynomial time procedure for the implication problem for a fragment of $\mathcal{DLFD}_{\text{reg}}$ that excludes negation, disallows defined concepts, and requires ordering concepts that occur in terminologies to satisfy a syntactic condition similar to the *regularity* condition in [10].

References

1. Marianne Baudinet, Jan Chomicki, and Pierre Wolper. Constraint-generating dependencies. *J. Comput. Syst. Sci.*, 59(1):94–115, 1999.
2. Joachim Biskup and Torsten Polle. Decomposition of Database Classes under Path Functional Dependencies and Onto Constraints. In *Foundations of Information and Knowledge Systems*, pages 31–49, 2000.
3. Joachim Biskup and Torsten Polle. Adding inclusion dependencies to an object-oriented data model with uniqueness constraints. *Acta Informatica*, 39:391–449, 2003.
4. Alexander Borgida and Grant Weddell. Adding Uniqueness Constraints to Description Logics (Preliminary Report). In *International Conference on Deductive and Object-Oriented Databases*, pages 85–102, 1997.
5. Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Identification Constraints and Functional Dependencies in Description Logics. In *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 155–160, 2001.
6. David DeHaan, David Toman, and Grant Weddell. Rewriting Aggregate Queries using Description Logics. In *Description Logics 2003*, pages 103–112. CEUR-WS vol.81, 2003.
7. Seymour Ginsburg and Richard Hull. Order Dependency in the Relational Model. *TCS*, 26:149–195, 1983.
8. Minoru Ito and Grant Weddell. Implication Problems for Functional Constraints on Databases Supporting Complex Objects. *Journal of Computer and System Sciences*, 49(3):726–768, 1994.
9. Vitaliy L. Khizder, David Toman, and Grant Weddell. Reasoning about Duplicate Elimination with Description Logic. In *Rules and Objects in Databases (DOOD, part of CL'00)*, pages 1017–1032, 2000.
10. Vitaliy L. Khizder, David Toman, and Grant Weddell. On Decidability and Complexity of Description Logics with Uniqueness Constraints. In *International Conference on Database Theory ICDT'01*, pages 54–67, 2001.

11. Carsten Lutz, Carlos Areces, Ian Horrocks, and Ulrike Sattler. Keys, Nominals, and Concrete Domains. In *Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 349–354, 2003.
12. Carsten Lutz and Maja Milicic. Description Logics with Concrete Domains and Functional Dependencies. In *Proc. European Conference on Artificial Intelligence (ECAI)*, pages 378–382, 2004.
13. Resource Description Framework (RDF). http://www.w3.org/RDF/.
14. Ulrike Sattler, Diego Calvanese, and Ralf Molitor. Relationships with other formalisms. In *The Description Logic Handbook: Theory, Implementation, and Applications*, chapter 4, pages 137–177. Cambridge University Press, 2003.
15. SPARQL Query Language for RDF. http://www.w3.org/TR/rdf-sparql-query/.
16. Lubomir Stanchev and Grant Weddell. Index Selection for Embedded Control Applications using Description Logics. In *Description Logics 2003*, pages 9–18. CEUR-WS vol.81, 2003.
17. David Toman and Grant Weddell. On Attributes, Roles, and Dependencies in Description Logics and the Ackermann Case of the Decision Problem. In *Description Logics 2001*, pages 76–85. CEUR-WS vol.49, 2001.
18. David Toman and Grant Weddell. Attribute Inversion in Description Logics with Path Functional Dependencies. In *Description Logics 2004*, pages 178–187. CEUR-WS vol.104, 2004.
19. David Toman and Grant Weddell. On Reasoning about Structural Equality in XML: A Description Logic Approach. *Theoretical Computer Science*, 336(1):181–203, 2005. doi:10.1016/j.tcs.2004.10.036.
20. David Toman and Grant Weddell. On the Interaction between Inverse Features and Path-functional Dependencies in Description Logics. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 603–608, 2005.
21. David Toman and Grant Weddell. On Keys and Functional Dependencies as First-Class Citizens in Description Logics. In *Proc. of the Third Int. Joint Conf. on Automated Reasoning (IJCAR)*, pages 647–661, 2006.
22. Web Ontology Language (OWL). http://www.w3.org/2004/OWL/.
23. Grant Weddell. A Theory of Functional Dependencies for Object Oriented Data Models. In *International Conference on Deductive and Object-Oriented Databases*, pages 165–184, 1989.
24. Grant Weddell. Reasoning about Functional Dependencies Generalized for Semantic Data Models. *TODS*, 17(1):32–64, 1992.