

Reliable Reference for a DL Knowledge Base under Data Update

Enamul Haque¹, David Toman¹ and Grant Weddell¹

¹*Cheriton School of Computer Science, University of Waterloo, 200 University Ave W., Waterloo, ON N2L 3G1, Canada*

Abstract

Earlier work has shown how Russell’s notion of proper definite descriptions can be captured as referring expressions: concept descriptions that are known to be singular. Concept descriptions are a more general and transparent way of communicating answers to queries. In general, how long such answers can be relied on will depend on how long facts about entities are not altered via data update, that is, on revisions to the underlying ABox of a KB. In this paper, we show how a simple variety of dynamic constraints can be added to a KB to ensure user specified durations on how long query answers must be able to be relied on.

Keywords

referring expressions, data update, dynamic constraints

1. Introduction


A knowledge base \mathcal{K} expressed in terms of a *description logic* (DL) will consist of a TBox, a finite set of *subsumptions* characterizing information relevant to the underlying domain of an application, and an ABox, a finite set of *assertions* that introduce specific domain facts. In this paper, we consider where \mathcal{K} is expressed in a dialect of the FunDL family of DLs [1] for which logical consequence of subsumptions and assertions in \mathcal{K} is computationally tractable. FunDL dialects are feature logics that replace *roles*, arbitrary binary relations, with *features*, arbitrary partial functions, and include a means of expressing subsumptions that capture a variety of equality generating dependencies via a concept constructor called a *path functional dependency* (PFD).

Examples of a TBox and ABox defining a knowledge base \mathcal{K} that are expressed in our DL are given in Figure 1 for a hypothetical university domain about STUDENTS, EMPLOYEES, BUILDINGS, and so on. Here, room1, "Davis Center" and 5678 are examples of so-called *individual names*, with the latter two also called *literal values* such as strings and integers. The subsumptions use PFDs (underlined) that will ensure, for example, that no two rooms will have a unique combination of a room number and the name of the building in which they reside. This ensures any structure of \mathcal{K} will have the same interpretation for each of the following two concepts that employ the nominal concept constructor:

$$\{\text{room1}\}, \text{ and } \text{ROOM} \sqcap \exists rnum. \{5678\} \sqcap \exists in.name. \{\text{"Davis Center"}\}. \quad (1)$$

The second concept is an example of a *referring expression* as introduced in [2], or, as Russell would say, a *proper definite description*.¹ This earlier work introduced the notion of a *referring expression type* (Rt) for specifying possible referring expressions for individuals. The syntax for an Rt was adapted in [4] for specifying concepts intended to serve as more transparent and readable referring expressions for individuals that are expressed as concepts. Based on this syntax, an Rt “generating” this referring expression for room1 in our university domain is as follows:

$$\text{ROOM} \sqcap \exists rnum. \langle ? \rangle \sqcap \exists in.name. \langle ? \rangle$$

 DL 2025: 38th International Workshop on Description Logics, September 3–6, 2025, Opole, Poland

 enamul.haque@uwaterloo.ca (E. Haque); david@uwaterloo.ca (D. Toman); gweddell@uwaterloo.ca (G. Weddell)



© 2025 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

¹For Russell, such expressions are well-formed formulae free in one variable that hold for exactly one element of a domain [3].

Note that occurrences of “ $\langle ? \rangle$ ” serve as placeholders for nominals, and that an Rt is therefore a pattern language for concepts.

One might expect that relying on the second referring expression in (1) to faithfully refer to room1 will always “work”, that room numbers, the buildings in which rooms reside and the names of buildings are permanent facts that will never change. However, now consider the following four concepts:

- (i) $\{\text{rob}\};$
 - (ii) $\text{PERSON} \sqcap \exists \text{ssn}.\{1234\};$
 - (iii) $\text{STUDENT} \sqcap \exists \text{snum}.\{321\};$ and
 - (iv) $\text{EMP} \sqcap \exists \text{enum}.\{77\}.$
- (2)

We will see that referring expressions (ii) and (iii) are in the language generated by an Rt defined as the following pattern:

$$(\text{STUDENT} \sqcap \exists \text{snum}.\langle ? \rangle) ; (\text{PERSON} \sqcap \exists \text{ssn}.\langle ? \rangle) \quad (3)$$

Note that the occurrence of “;” in this Rt expresses a preference for referring expression (iii) over referring expression (ii) in (2). Here again, any structure of \mathcal{K} will have the same interpretation for each of the concepts. However, in this case, referring to rob via referring expressions (iii) or (iv) will only work for as long as rob continues to be student and an employee, a circumstance that might no longer be true in some future update to the ABox.

It is this kind of effect that sensible data update can have on referring expressions that is our primary concern. In particular, we introduce a form of *dynamic constraint* that can be added to \mathcal{K} to ensure the efficacy of a referring expression.

To illustrate, if the facts that rob is a student and that his student number is 321 are implied by \mathcal{K} , consider where \mathcal{K} should also ensure that any data update on the ABox in which he continues to be a student must also preserve knowledge of this number. We augment the above mentioned FunDL dialect for capturing \mathcal{K} to incorporate a form of dynamic constraint to ensure this, in particular, a dynamic constraint of the form

$$(\text{STUDENT}, \text{snum}).$$

The constraint is dynamic in the sense that it fails to hold when also considering any data update to the ABox in which rob remains a student but in which knowledge of this student number is either removed or updated.

Our main result introduces the procedure $\text{ChooseRE}(a, C, \mathcal{K})$ where a is an individual name, C a concept expressing a “duration requirement” and \mathcal{K} a knowledge base that now includes dynamic constraints, such as the above, and an Rt . The procedure attempts to compute the most preferred referring expression in the language generated by Rt for referring to a that can be relied on for as long as a is known to be a C .

For example, assume the TBox and ABox of \mathcal{K} are as given in Figure 1, where it has additional dynamic constraints such as the above, and where it has the Rt given by (3). Then $\text{ChooseRE}(\text{rob}, \text{STUDENT}, \mathcal{K})$ would return referring expression (iii) in (2) and $\text{ChooseRE}(\text{rob}, \text{PERSON}, \mathcal{K})$ would return referring expression (ii). However, if the fact that Rob’s student number is 321 was not in the ABox, then the first call would also return referring expression (ii).

To summarize, our contributions revolve around the ChooseRE procedure. In particular, we introduce a variety of dynamic constraints relating to individuals and their feature values and incorporate such constraints in the computation of referring expressions for individuals that (a) are more general and transparent ways of communicating references to individuals, for example, in answers to queries, and (b) can be trusted to reliably refer for a specified duration.

The remainder of the paper is organized as follows. Section 2 provides the needed background relating to our FunDL dialect, to referring expressions, and to our new dynamic constraints. Procedure ChooseRE is then introduced in Section 3. In Section 4, we consider a number of additional issues relating to “unique name” possibilities, to counting, and to incremental ways to guarantee the existence of referring expressions. Summary comments are given in a final subsection.

$$\begin{aligned}
\text{TBox} = \{ & \text{ROOM} \sqsubseteq (\exists rnum.\top) \sqcap (\exists in.\text{BLD}) \sqcap \underline{\text{ROOM}} : rnum, in.name \rightarrow id, \\
& \text{BLD} \sqsubseteq (\exists name.\top) \sqcap \underline{\text{BLD}} : name \rightarrow id, \\
& \text{PERSON} \sqsubseteq (\exists ssn.\top) \sqcap (\exists lname.\top) \sqcap \underline{\text{PERSON}} : ssn \rightarrow id, \\
& \text{STUDENT} \sqsubseteq \text{PERSON} \sqcap (\exists snum.\top) \sqcap \underline{\text{STUDENT}} : snum \rightarrow id, \\
& \text{EMP} \sqsubseteq \text{PERSON} \sqcap (\exists enum.\top) \sqcap (\exists loc.\text{ROOM}) \sqcap \underline{\text{EMP}} : enum \rightarrow id \} \\
\text{ABox} = \{ & \text{ROOM}(\text{room1}), rnum(\text{room1}) = 5678, in(\text{room1}) = \text{bld1}, \\
& \text{BLD}(\text{bld1}), name(\text{bld1}) = \text{"Davis Center"}, \\
& \text{PERSON}(\text{rob}), ssn(\text{rob}) = 1234, lname(\text{rob}) = \text{"Smith"}, \\
& \text{STUDENT}(\text{rob}), snum(\text{rob}) = 321, \\
& \text{PERSON}(\text{robin}), ssn(\text{robin}) = 1234, lname(\text{robin}) = \text{"Smith"}, \\
& \text{EMP}(\text{robin}), enum(\text{robin}) = 77, loc(\text{robin}) = \text{room1} \}
\end{aligned}$$

Figure 1: UNIVERSITY SUBSUMPTIONS AND ASSERTIONS.

SYNTAX	SEMANTICS: DEFN OF “ $(\cdot)^{\mathcal{I}}$ ”
$C ::= \perp$	\emptyset
$ C : \text{Pf}_1, \dots, \text{Pf}_k \rightarrow \text{Pf}_0$	$\{x \mid \forall y. ((y \in C^{\mathcal{I}} \wedge (\bigwedge_{i=0}^k \{x, y\} \subseteq (\exists \text{Pf}_i.\top)^{\mathcal{I}}))$ $(\bigwedge_{i=1}^k \text{Pf}_i^{\mathcal{I}}(x) = \text{Pf}_i^{\mathcal{I}}(y))) \rightarrow (\text{Pf}_0^{\mathcal{I}}(x) = \text{Pf}_0^{\mathcal{I}}(y)))\}$
$ \top$	$\Delta^{\mathcal{I}}$
$ A$	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
$ \exists \text{Pf}.C$	$\{x \mid \exists y. (y \in C^{\mathcal{I}} \wedge \text{Pf}^{\mathcal{I}}(x) = y)\}$
$ C_1 \sqcap C_2$	$C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$
$ \{a\}$	$\{a^{\mathcal{I}}\}$
$ \exists f^{-1}.C$	$\{f^{\mathcal{I}}(x) \mid x \in C^{\mathcal{I}}\}$

Figure 2: SYNTAX AND SEMANTICS OF CONCEPT DESCRIPTIONS.

2. Definitions

We now formally define the artifacts introduced in our introductory comments, beginning with the definition of concepts and TBoxes for a member of the FunDL family of DLs with decidable complexity of logical consequence for subsumptions and for assertions. Recall that members of this family replace roles with partial functions, and that concepts also serve the role of referring expressions. Also note that we distinguish a countably infinite subset of individual names that are literal values such as strings or integers and for which we adopt the unique name assumption.

Definition 1 (FunDL Concepts, Referring Expressions, and TBoxes). *Let F , PC , IN , and D be respective countably infinite sets of feature names $\{f_1, f_2, \dots\}$, primitive concept names $\{A_1, A_2, \dots\}$, individual names $\{a_1, a_2, \dots\}$, and a countably infinite subset of IN that are literal values. A path expression is defined by the grammar “ $\text{Pf} ::= f. \text{Pf} \mid id$ ” for $f \in F$ and a concept by the grammar on the left-hand-side of Figure 2. Concepts generated by the second production are called path functional dependencies (PFDs).²*

A referring expression (*Re*) is a concept description parsed by the last six productions in Figure 2; these are intended to assert the existence of individuals with complex properties.

A subsumption is an expression of the form $C_1 \sqsubseteq C_2$, where the C_i are FunDL concepts parsed by the first six productions in Fig. 2. A terminology (*TBox*) \mathcal{T} consists of a finite set of subsumptions.

The semantics of concepts and path expressions is defined with respect to a structure $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is a domain of “individuals” including “literal values”, and where $\cdot^{\mathcal{I}}$ is an interpretation function that

²Recall that such concepts are the above-mentioned means of capturing equality generating dependencies.

fixes the interpretations of primitive concepts A to be subsets of $\Delta^{\mathcal{I}}$ and primitive features f to be partial functions $f^{\mathcal{I}} : \Delta^{\mathcal{I}} \rightarrow \Delta^{\mathcal{I}}$. The interpretation function also satisfies the unique name assumption (UNA) for literal values, that is, that $(a_1)^{\mathcal{I}} \neq (a_2)^{\mathcal{I}}$ for any pair of distinct a_1 and a_2 in D . The interpretation is extended in the natural way to path expressions: $id^{\mathcal{I}} = \lambda x.x$, $(f.Pf)^{\mathcal{I}} = Pf^{\mathcal{I}} \circ f^{\mathcal{I}}$; and to concept descriptions as indicated on the right-hand-side of Figure 2.

A structure \mathcal{I} satisfies a subsumption $C_1 \sqsubseteq C_2$ if $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$, and is a model of a TBox \mathcal{T} if it satisfies all subsumptions in \mathcal{T} . A subsumption is a logical consequence of a TBox, written $\mathcal{T} \models (C_1 \sqsubseteq C_2)$, when every model of \mathcal{T} also satisfies $C_1 \sqsubseteq C_2$.

Given a TBox \mathcal{T} , a referring expression C is singular with respect to \mathcal{T} if $|C^{\mathcal{I}}| \leq 1$ for any model \mathcal{I} of \mathcal{T} . \square

Unfortunately, an unrestricted use of the first six concept constructors in Fig. 2 in TBox subsumptions still leads to undecidability of KB consistency and logical implication questions [5]. To regain decidability, all PFDs in the TBox must appear on right-hand sides of subsumptions and must conform to the following forms:

1. $C : Pf_1, \dots, Pf_i.Pf_i, \dots, Pf_k \rightarrow Pf$ or
2. $C : Pf_1, \dots, Pf.f_1, \dots, Pf_k \rightarrow Pf.f_2$

With these restrictions, reasoning tasks become complete for EXPTIME. Further restrictions are needed to obtain PTIME reasoning algorithms for the above tasks. The most general form of such restrictions (to date) has been developed in [6].

Referring expression types are now defined. Recall from our introductory comments that these were first introduced in [2], and that the version presented here is from later work in [4] which adapts earlier syntax to conform with referring expressions expressed as concepts. In this version, we have made a minor revision to enable such types to distinguish the case in which “feature paths” lead more specifically to literal values. The discussion that follows on how a referring expression type can be diagnosed at “compile time” to determine if all generated referring expressions are singular w.r.t. a given TBox is also from [4]. This entails the introduction of a couple of useful auxiliary functions that is used in later sections.

Definition 2 (FunDL Referring Expression Types). A referring expression type (Rt) is defined by the following grammar:³

$$Rt ::= A \mid \{?\} \mid \langle ? \rangle \mid \exists Pf.Rt \mid Rt \sqcap Rt \mid Rt ; Rt$$

We define the language of referring expressions inhabiting Rt , $\mathcal{L}(Rt)$, as follows:

$$\begin{aligned} \mathcal{L}(A) &= \{A\} \\ \mathcal{L}(\{?\}) &= \{\{a\} \mid a \in \text{IN}\} \\ \mathcal{L}(\langle ? \rangle) &= \{\{a\} \mid a \in D\} \\ \mathcal{L}(\exists Pf.Rt) &= \{\exists Pf.C \mid C \in \mathcal{L}(Rt)\} \\ \mathcal{L}(Rt_1 \sqcap Rt_2) &= \{C_1 \sqcap C_2 \mid C_1 \in \mathcal{L}(Rt_1) \text{ and } C_2 \in \mathcal{L}(Rt_2)\} \\ \mathcal{L}(Rt_1 ; Rt_2) &= \mathcal{L}(Rt_1) \cup \mathcal{L}(Rt_2) \end{aligned}$$

Also, we write $\text{Norm}(Rt)$ to refer to an exhaustive application of the following rewrite rules to Rt :

$$\begin{aligned} Rt \sqcap (Rt_1 ; Rt_2) &\mapsto Rt \sqcap Rt_1 ; Rt \sqcap Rt_2 \\ (Rt_1 ; Rt_2) \sqcap Rt &\mapsto Rt_1 \sqcap Rt ; Rt_2 \sqcap Rt \\ \exists Pf.(Rt_1 ; Rt_2) &\mapsto \exists Pf.Rt_1 ; \exists Pf.Rt_2 \end{aligned}$$

\square

³This is a pattern language obtained by abstracting nominals in referring expressions, and by admitting a final production to express *preference* among referring expressions [2]. Also note that such a preference only becomes an issue when more than one referring expression for an individual is possible in an ABox, in particular, in defining our ChooseRE procedure in the next section.

The definition of $\text{Norm}(Rt)$ is a simple variant of referring expression type normalization defined in [2], and the following are consequences: (1) $\mathcal{L}(Rt) = \mathcal{L}(\text{Norm}(Rt))$, and (2) all *preference operators* (“;”) are at the top level of $\text{Norm}(Rt)$. We call the maximal “;”-free parts of $\text{Norm}(Rt)$ *preference-free components*.

Given a TBox \mathcal{T} and a referring expression type Rt , the following auxiliary functions will enable a static test for singularity of referring expressions in $\mathcal{L}(Rt)$:

$$\begin{array}{ll} \text{Con}(A) = A & \text{Pfs}(A) = \{ \} \\ \text{Con}(\{?\}) = \top & \text{Pfs}(\{?\}) = \{id\} \\ \text{Con}(\langle?\rangle) = \top & \text{Pfs}(\langle?\rangle) = \{id\} \\ \text{Con}(\exists Pf'. Rt) = \exists Pf'. \text{Con}(Rt) & \text{Pfs}(\exists Pf'. Rt) = \{Pf' . Pf \mid Pf \in \text{Pfs}(Rt)\} \\ \text{Con}(Rt_1 \sqcap Rt_2) = \text{Con}(Rt_1) \sqcap \text{Con}(Rt_2) & \text{Pfs}(Rt_1 \sqcap Rt_2) = \text{Pfs}(Rt_1) \cup \text{Pfs}(Rt_2) \end{array}$$

The functions extract a FunDL concept and a set of path expressions leading to nominals from a preference-free referring expression type. A straightforward variation of the ideas presented in [2], Theorem 20, yields the following formulation of the static test:

Theorem 1 (from [4]). *Let \mathcal{T} be a TBox and Rt a referring expression type. Then all referring expressions in $\mathcal{L}(Rt)$ are singular if and only if for every preference-free component Rt' of $\text{Norm}(Rt)$:*

$$\mathcal{T} \models \text{Con}(Rt') \sqsubseteq \text{Con}(Rt') : \text{Pfs}(Rt') \rightarrow id.$$

□

The definition of our dynamic constraints now follows and includes a definition of ABoxes and of data update. Note that a finite set of dynamic constraints and a (single) referring expression type are now included as part of the definition of a knowledge base and that we have taken a very general view of what constitutes data update: replacing an ABox with an entirely new ABox. More discussion will follow.

Definition 3 (FunDL ABoxes, Dynamic Constraints and Data Update). *An assertion box (ABox) \mathcal{A} consists of a finite set of assertions of the form $C(a_1)$, $a_1 = a_2$, or $f(a_1) = a_2$, where C is a concept and the a_i are individual names.*

A dynamic constraint has the form (C, Pf) , and we write \mathcal{W} to refer to a WBox, a finite set of dynamic constraints.

A knowledge base \mathcal{K} is a four-tuple $(\mathcal{T}, \mathcal{A}, \mathcal{W}, Rt)$.

A structure \mathcal{I} satisfies \mathcal{A} when it satisfies each assertion in \mathcal{A} , that is, when $(a_1)^{\mathcal{I}} \in C^{\mathcal{I}}$, $(a_1)^{\mathcal{I}} = (a_2)^{\mathcal{I}}$ and $f^{\mathcal{I}}((a_1)^{\mathcal{I}}) = (a_2)^{\mathcal{I}}$. $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{W}, Rt)$ is consistent if there exists a structure over \mathcal{K} , that is, that satisfies \mathcal{T} and \mathcal{A} .

A data update is an ABox \mathcal{A} , and qualifies as an update on $\mathcal{K} = (\mathcal{T}, \mathcal{A}', \mathcal{W}, Rt)$ if \mathcal{K} is consistent, $\mathcal{K}' = (\mathcal{T}, \mathcal{A}, \mathcal{W}, Rt)$, also written \mathcal{K}/\mathcal{A} , is also consistent, and if \mathcal{K} and \mathcal{A} also satisfy each dynamic constraint (C, Pf) in \mathcal{W} , written $(\mathcal{K}, \mathcal{A}) \models (C, Pf)$. This holds when:

$$\begin{array}{l} \text{for any } \mathcal{I}_1 \text{ over } \mathcal{K}, \mathcal{I}_2 \text{ over } \mathcal{K}/\mathcal{A} \text{ and individuals } a_1 \text{ and } a_2, \text{ if } (a_1)^{\mathcal{I}_1} \in C^{\mathcal{I}_1}, \\ (a_1)^{\mathcal{I}_2} \in C^{\mathcal{I}_2} \text{ and } Pf^{\mathcal{I}_1}((a_1)^{\mathcal{I}_1}) = (a_2)^{\mathcal{I}_1} \text{ then } Pf^{\mathcal{I}_2}((a_1)^{\mathcal{I}_2}) = (a_2)^{\mathcal{I}_2}. \end{array} \quad (4)$$

More generally, we write $\mathcal{K} \models (C, Pf)$ when $(\mathcal{K}, \mathcal{A}) \models (C, Pf)$ for any data update \mathcal{A} on \mathcal{K} .

□

Our notion of a data update is based on the notion of a transaction on a relational database that updates only the contents of tables and is consistency preserving, that is consists of inserts, updates and deletes on a given collection of tables for which all integrity constraints continue to hold. Also, in earlier work, we have considered where an Rt was attached to each free variable of a query [2] and, more recently, where an Rt is attached instead to primitive concepts in the context of a DL-based knowledge base [4]. It turns out in the latter case that a single Rt obtained by using “;” to “catenate”

those attached to some primitive concept, thus establishing a global preference for how to refer to any object, suffices.

Observe that dynamic constraints cannot disqualify the ABox of a consistent knowledge base \mathcal{K} from also qualifying as a data update on \mathcal{K} , nor can revising such constraints of a consistent knowledge base lead to its inconsistency. This leads immediately to the following:

Theorem 2. *Let $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{W}, Rt)$ be a consistent knowledge base and C a subsumption or assertion. Then $\mathcal{K} \models C$ iff $(\mathcal{T}, \mathcal{A}, \{\}, Rt) \models C$, that is, admitting dynamic constraints in a knowledge base are a conservative extension w.r.t. logical consequence of subsumptions or assertions.* \square

3. Ensuring Durations for Referring Expressions

We now define procedure $\text{ChooseRE}(a, C, \mathcal{K})$ for computing more transparent and readable referring expressions for an individual a in knowledge base \mathcal{K} that can be relied on for duration C , that is, for as long as a is known to be an instance of concept C . The procedure uses the definition of ToRE given in [4] for computing a referring expression that works “in the here and now” for a given “;”-free Rt' . In particular, ToRE is called in an iterative fashion on the sequence of such Rt' in $\text{Norm}(Rt)$ until finding a referring expression that also satisfies a subsumption relating to C and Rt' . In addition, durability requires that \mathcal{K} itself satisfies a durability condition.

Note that, unlike the case in [4], it is now possible that different referring expressions are obtained for the same individual name due to alternative choices for C , that is, for durations. Again, more discussion will follow.

Definition 4 (Choosing a Referring Expression). *Let a be an individual name, C a concept and $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{W}, Rt)$ a consistent knowledge base, and assume $\text{Norm}(Rt) = Rt_1; \dots; Rt_k$. \mathcal{K} is durable when the following holds:*

$$\mathcal{K} \models (\text{Con}(Rt_i), \text{Pf}), \text{ for } 1 \leq i \leq k \text{ and all } \text{Pf} \in \text{Pfs}(Rt_i). \quad (5)$$

We define $\text{ToRE}(a, Rt_i, \mathcal{K})$ to be the result of the following recursive definition of $\text{ToRE}(a, Rt_i, id, \mathcal{K})$ on the structure of Rt_i :

$$\begin{aligned} \text{ToRE}(a, A, \text{Pf}, \mathcal{K}) &= A \text{ if } \mathcal{K} \models a : \exists \text{Pf}. A, \text{ undefined otherwise;} \\ \text{ToRE}(a, \{\}, \text{Pf}, \mathcal{K}) &= \{a'\} \text{ if } \mathcal{K} \models a : \exists \text{Pf}. \{a'\} \text{ for some } a' \in \text{IN}, \text{ undefined otherwise;} \\ \text{ToRE}(a, \langle ? \rangle, \text{Pf}, \mathcal{K}) &= \{a'\} \text{ if } \mathcal{K} \models a : \exists \text{Pf}. \{a'\} \text{ for some } a' \in \text{D}, \text{ undefined otherwise;} \\ \text{ToRE}(a, \exists \text{Pf}'. Rt, \text{Pf}, \mathcal{K}) &= \exists \text{Pf}'. \text{ToRE}(a, Rt, \text{Pf} \cdot \text{Pf}', \mathcal{K}); \text{ and} \\ \text{ToRE}(a, Rt_1 \sqcap Rt_2, \text{Pf}, \mathcal{K}) &= \text{ToRE}(a, Rt_1, \text{Pf}, \mathcal{K}) \sqcap \text{ToRE}(a, Rt_2, \text{Pf}, \mathcal{K}) \text{ if both are defined,} \\ &\text{undefined otherwise.} \end{aligned}$$

We write $\text{ChooseRE}(a, C, \mathcal{K})$ to return a concept C' for the least $i \leq k$ for which the following hold, and to be undefined otherwise:

1. $\mathcal{K} \models C \sqsubseteq \text{Con}(Rt_i)$,
2. $\text{ToRE}(a, Rt_i, id, \mathcal{K})$ is defined and returns C' . \square

See earlier work [7, 8] for more effective ways of computing the second and third cases of ToRE by appealing to logical consequence in FunDL knowledge bases based on a binary search that assumes access to a total ordering of individual names IN. This earlier work also shows how standard classification can be employed to compute the first case of ToRE .

Building on our university domain, consider where the TBox and ABox for \mathcal{K} are as given in Figure 1. Also assume the dynamic constraints and referring expression type for \mathcal{K} are as given in Figure 3. Observe that the *name* of a person and the *location* of an employee office are really the only kinds

$$\begin{aligned}
\text{WBox} &= \{ (\text{ROOM}, rnum), (\text{ROOM}, in), (\text{BLD}, name), \\
&\quad (\text{PERSON}, ssn), (\text{STUDENT}, snum), (\text{EMP}, enum) \} \\
Rt &= \text{ROOM} \sqcap \exists rnum.\langle ? \rangle \sqcap \exists in.name.\langle ? \rangle ; \\
&\quad \text{BLD} \sqcap \exists name.\langle ? \rangle ; \\
&\quad \text{STUDENT} \sqcap \exists snum.\langle ? \rangle ; \text{EMP} \sqcap \exists enum.\langle ? \rangle ; \text{PERSON} \sqcap \exists ssn.\langle ? \rangle
\end{aligned}$$

Figure 3: UNIVERSITY DYNAMIC CONSTRAINTS AND REFERRING EXPRESSION TYPE.

of facts that can updated. Then the following lists examples of calls to ChooseRE and the referring expression returned as a consequence:

$$\begin{aligned}
\text{ChooseRE}(\text{rob}, \text{STUDENT}, \mathcal{K}) &\rightarrow \text{STUDENT} \sqcap \exists snum.\{321\} \\
\text{ChooseRE}(\text{rob}, \text{PERSON}, \mathcal{K}) &\rightarrow \text{PERSON} \sqcap \exists ssn.\{1234\} \\
\text{ChooseRE}(\text{room1}, \text{ROOM}, \mathcal{K}) &\rightarrow \text{ROOM} \sqcap \exists rnum.\{5678\} \sqcap \exists in.name.\{\text{"Davis Center"}\} \\
\text{ChooseRE}(\text{rob}, \text{EMP} \sqcap \exists loc.in.name.\{\text{"David Center"}\}, \mathcal{K}) &\rightarrow \text{EMP} \sqcap \exists enum.\{77\}
\end{aligned}$$

Note that the first three are as reported in our introductory comments. The fourth shows another referring expression for rob that is requested to last for as long as he is an employee with an office located in the David Center building.

Theorem 3. *Let a be an individual name, C_1 a concept and $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{W}, Rt)$ a consistent knowledge base that is durable and for which all referring expressions in $\mathcal{L}(Rt)$ are singular w.r.t. \mathcal{T} . If $\text{ChooseRE}(a, C_1, \mathcal{K})$ is defined and returns concept C_2 , then $C_2^{\mathcal{I}_1} = C_2^{\mathcal{I}_2} = \{(a)^{\mathcal{I}_1}\}$ for any \mathcal{I}_1 over \mathcal{K} , any data update \mathcal{A}' on \mathcal{K} and any \mathcal{I}_2 over \mathcal{K}/\mathcal{A}' .*

Proof. (sketch) By induction on the structure of the concept C_2 generated by $\text{ToRE}(a, Rt_i, id, \mathcal{K})$ for some Rt_i . The invariant of the structural induction is $\{\text{Pf} . \text{Pf}' \mid \text{Pf}' \in \text{Pfs}(Rt_i)\} \subseteq \text{Pfs}(Rt_i)$ where Rt and Pf are the second and third arguments of the recursive calls to ToRE in Definition 4. Then in the first base case $\mathcal{K} \models a : \exists \text{Pf}.A$ holds as $\text{Con}(Rt_i) \sqsubseteq \exists \text{Pf}.A$ and in the second and third case we have $\text{Pf} \in \text{Pfs}(Rt_i)$ and thus, due to the requirement $\mathcal{K} \models (\text{Con}(Rt_i), \text{Pf})$ for all $\text{Pf} \in \text{Pfs}(Rt_i)$ and condition (4), we get Pf reaches the same constant a' starting from a in \mathcal{I}_2 as it reached in \mathcal{I}_1 (when constructing C_2 using ToRE). \square

Recall that logical consequence for subsumptions is decidable. Thus, by Theorem 2, the only outstanding computational issue with $\text{ChooseRE}(a, C_1, \mathcal{K})$ concerns logical consequence for dynamic constraints, that is, to determine if $\mathcal{K} \models (C, \text{Pf})$ for some \mathcal{K} , C and Pf . To this end, we define two inference axioms:

$$\frac{\mathcal{K} \models (C_1, \text{Pf}), \mathcal{K} \models C_2 \sqsubseteq C_1}{\mathcal{K} \models (C_2, \text{Pf})} \quad (6)$$

$$\frac{\mathcal{K} \models (C_1, \text{Pf}_1), \mathcal{K} \models C_1 \sqsubseteq \exists \text{Pf}_1.C_2, \mathcal{K} \models (C_2, \text{Pf}_2)}{\mathcal{K} \models (C_1, \text{Pf}_1 . \text{Pf}_2)} \quad (7)$$

A sound procedure to decide if $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{W}, Rt) \models (C, \text{Pf})$ based on these axioms can operate by first checking if C is not satisfiable or if Pf is id and returning true if this is the case. If C is satisfiable and Pf is not id , the procedure can then proceed in an iterative manner on the sequence of feature names occurring in Pf . In particular, if Pf has the form $f . \text{Pf}'$, check if there exists a C_1 and C_2 where $(C_1, f) \in \mathcal{W}$, $\mathcal{K} \models C \sqsubseteq C_1$ and $\mathcal{K} \models C_1 \sqsubseteq \exists f.C_2$, and then recurse on deciding if $\mathcal{K} \models (C_2, \text{Pf}')$ if Pf' is not id .

4. Discussion

4.1. On UNA and Counting

As with individual names that are not literal values for which the UNA applies, referring expressions can in principle co-refer to the same object in a structure for a given \mathcal{K} . This was indeed the case in our introductory example (2):

$$\{\text{rob}\} \quad \text{PERSON} \sqcap \exists \text{ssn}.\{1234\} \quad \text{STUDENT} \sqcap \exists \text{snum}.\{321\} \quad \text{EMP} \sqcap \exists \text{enum}.\{77\}$$

On the other hand, literal values (elements of \mathcal{D}) will be distinct since we have assumed the UNA for literal values. Hence, referring expressions in $\mathcal{L}(\langle ? \rangle)$ will inherit this property. In the following we show how the UNA can be lifted to more complex referring expression types. In this way we ensure that referring expressions that inhabit these types also obey this lifted variant of UNA: syntactically distinct referring expressions must always refer to distinct domain elements. The following condition on referring expression types lifts UNA to more complex referring expression types:

- Let $\text{Norm}(Rt) = Rt_1; \dots; Rt_k$ be “ $\{?\}$ ”-free referring expression type such that $\mathcal{K} \models \text{Con}(Rt_i) \sqcap \text{Con}(Rt_j) \sqsubseteq \perp$ for all $i < j \leq k$. Then $\mathcal{L}(Rt)$ obeys lifted UNA.

Note that, e.g., stronger lower bounds for counting are enabled by detecting lifted UNA.

4.2. On Guaranteeing the Existence of Referring Expressions

The procedure for choosing a referring expression in Definition 4 may fail to return an appropriate referring expression. We focus on the situation in which the reason for failure is entirely due to where $\text{ToRE}(a, Rt_i, \mathcal{K})$ is not defined, i.e., where \mathcal{K} does not entail sufficiently many *facts* about the individual a in question.

One can strengthen definition (4) of the semantics of a dynamic constraint (C, Pf) to ensure, whenever $\mathcal{K} \models C(a)$ become true, $\text{ToRE}(a, Rt_i, \mathcal{K})$ will then succeed and produces a referring expression in $\mathcal{L}(Rt_i)$ that refers to a with the duration C . This additional requirement for (C, Pf) is as follows:

$$\text{for any } \mathcal{I}_1 \text{ over } \mathcal{K}, \mathcal{I}_2 \text{ over } \mathcal{K}/\mathcal{A} \text{ and individual } a_1, \text{ if } (a_1)^{\mathcal{I}_1} \notin C^{\mathcal{I}_1} \text{ and } (a_1)^{\mathcal{I}_2} \in C^{\mathcal{I}_2} \text{ then there is } a_2 \text{ such that } \text{Pf}^{\mathcal{I}_2}((a_1)^{\mathcal{I}_2}) = (a_2)^{\mathcal{I}_2}. \quad (8)$$

Note that we still require \mathcal{K} to be durable, in particular that the following holds:

$$\mathcal{K} \models (\text{Con}(Rt_i), \text{Pf}), \text{ for } 1 \leq i \leq k \text{ and all } \text{Pf} \in \text{Pfs}(Rt_i),$$

and that these now satisfy both conditions (4) and (8). In addition, we need to require that for $\text{Pf} \in \text{LitPfs}(Rt_i)$ the constant a_2 in (8) is in \mathcal{D} (i.e., the path ends in a literal as prescribed by Rt_i). The auxiliary function LitPfs is defined as follows:

$$\begin{aligned} \text{LitPfs}(\mathbf{A}) &= \{ \} \\ \text{LitPfs}(\{?\}) &= \{ \} \\ \text{LitPfs}(\langle ? \rangle) &= \{id\} \\ \text{LitPfs}(\exists \text{Pf}' . Rt) &= \{ \text{Pf}' . \text{Pf} \mid \text{Pf} \in \text{LitPfs}(Rt) \} \\ \text{LitPfs}(Rt_1 \sqcap Rt_2) &= \text{LitPfs}(Rt_1) \cup \text{LitPfs}(Rt_2) \end{aligned}$$

With this stronger requirement, $\text{ToRE}(a, Rt_i, \mathcal{K})$ always returns an appropriate referring expression for any a that will persist as long as $C(a)$ holds. Hence, evolving \mathcal{K} via data updates starting from an empty ABox guarantees the existence of referring expressions for any individual for which such an expression can exist.

4.3. Summary Comments

We have developed dynamic constraints—constraints on allowed ABox changes—that make referring expressions durable relative to concept membership of the objects they identify. This way we can guarantee that, e.g., the value of the feature *snum* can be used to identify PERSONs as long as they are STUDENTs, but ceases to be reliable when a PERSON ceases to be a STUDENT (and we have to revert to other ways of identifying such objects). This durability of referring expressions relative to concept membership has many applications, for example, when one considers physical representation of such objects in multi-level storage systems (such as caches). We also introduced conditions under which the existence of appropriate referring expressions is guaranteed for all objects belonging to a given concept description. Last, we studied how UNA for constant symbols can be lifted to complex referring expressions.

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

- [1] S. McIntyre, D. Toman, G. E. Weddell, FunDL - A family of feature-based description logics, with applications in querying structured data sources, in: *Description Logic, Theory Combination, and All That - Essays Dedicated to Franz Baader on the Occasion of His 60th Birthday*, 2019, pp. 404–430.
- [2] A. Borgida, D. Toman, G. Weddell, On referring expressions in query answering over first order knowledge bases, in: *Proc. Principles of Knowledge Representation and Reasoning, KR 2016*, 2016, pp. 319–328.
- [3] B. Russell, On denoting, *Mind* 14 (1905) 479–493. URL: <http://www.jstor.org/stable/2248381>.
- [4] A. Borgida, E. Franconi, D. Toman, G. E. Weddell, Understanding document data sources using ontologies with referring expressions, in: *AI 2022: Advances in Artificial Intelligence*, volume 13728 of *LNCS*, Springer, 2022, pp. 367–380.
- [5] D. Toman, G. E. Weddell, On Keys and Functional Dependencies as First-Class Citizens in Description Logics, *J. Aut. Reasoning* 40 (2008) 117–132.
- [6] S. McIntyre, A. Borgida, D. Toman, G. E. Weddell, On limited conjunctions and partial features in parameter-tractable feature logics, in: *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019*, 2019, pp. 2995–3002.
- [7] J. Pound, D. Toman, G. E. Weddell, J. Wu, Query algebra and query optimization for concept assertion retrieval, in: V. Haarslev, D. Toman, G. E. Weddell (Eds.), *Proceedings of the 23rd International Workshop on Description Logics (DL 2010)*, volume 573 of *CEUR Workshop Proceedings*, 2010.
- [8] J. Pound, D. Toman, G. E. Weddell, J. Wu, An assertion retrieval algebra for object queries over knowledge bases, in: T. Walsh (Ed.), *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, 2011, IJCAI/AAAI, 2011*, pp. 1051–1056.