Computing Concept Referring Expressions for Queries on Horn $\mathcal{ALC}$ Ontologies

Moritz Illich , **Birte Glimm**

Institute of Artificial Intelligence, Ulm University, Germany

moritz.illich@uni-ulm.de, birte.glimm@uni-ulm.de

Abstract

Classical instance queries over an ontology only consider explicitly named individuals. *Concept referring expressions (CREs)* also allow for returning answers in the form of concepts that describe implicitly given individuals in terms of their relation to an explicitly named one. Existing approaches, e.g., based on tree automata, can neither be integrated into state-of-the-art OWL reasoners nor are they directly amenable for an efficient implementation. To address this, we devise a novel algorithm that uses highly optimized OWL reasoners as a black box. In addition to the standard criteria of singularity and certainty for CREs, we devise and consider the criterion of uniqueness of CREs for Horn $\mathcal{ALC}$ ontologies. The evaluation of our prototypical implementation shows that computing CREs for the most general concept ($\top$) can be done in less than one minute for ontologies with thousands of individuals and concepts.

1 Introduction

Description Logics (DLs) are a family of logic-based representation formalisms that provide the logical underpinning of the well-known Web Ontology Language (OWL) [Hitzler *et al.*, 2009]. The expressed knowledge is typically separated into terminological (aka TBox or schema) and assertional knowledge (aka ABox or facts). The TBox describes the relationships between concepts (representing sets of individuals with common characteristics) as well as roles (specifying the relationships between pairs of individuals). The ABox defines concrete instances of concepts and roles. Automated reasoning systems derive implicit consequences of the explicitly stated information, e.g., when answering queries for concept instances. Classically, answers to such queries consist of individual names that are used in the knowledge base. Consider, for example, a knowledge base that states that *every person has a mother who is a person* and that *John is a person* (in DL syntax: *Person* $\sqsubseteq$ $\exists hasMother.Person$ and *Person*(*john*), where *Person* is a concept, *hasMother* is a role, and *john* is an individual). Obviously, *john* is an answer to the query for instances of the concept *Person* and

such names are also called *referring expressions* as they refer to elements in the models of the knowledge base. In each model of the knowledge base, however, there is also an element that represents *the mother of John*. This *anonymous* element can be described by a *concept referring expression* (CRE) [Areces *et al.*, 2008; Borgida *et al.*, 2016; Toman and Weddell, 2019], i.e., by a concept that describes an anonymous element w.r.t. a named individual. For our example, we can use the CRE *the person who is the mother of John* (as DL concept: *Person* $\sqcap$ $\exists hasMother^-.\{john\}$, where *hasMother*$^-$ denotes the inverse relation *is mother of* of *hasMother*). The reasoning task of *generalized instance retrieval* refers to computing entailed named and anonymous answers in the form of CREs for a concept instance query, which allows for more comprehensive query results.

Usually, CREs are meant to identify a *single* element (e.g., *John's mother* and not some set of elements as, e.g., the concept *Person* itself). While some approaches for computing CREs rely on functionality of roles and of role paths in order to guarantee singularity, e.g., [Borgida *et al.*, 2016], we do not make this restriction as also the recent work of Toman and Weddell [2019]. Another challenge when computing CREs is describing *all* entailed answers to (concept instance) queries. While the approach of Toman and Weddell based on tree automata addresses this problem, it is not well-suited for an efficient implementation. Moreover, the applied constraints that ensure the construction of only singular CREs are actually too strict, thus neglecting query answers in some cases, while an explicit prevention of duplicate answers referring to different, but semantically equivalent, CREs is also not provided. In this paper, we address these issues and present the first practical method for generalized instance retrieval that uses highly optimized reasoners as a black box for computing CREs.

At first, we introduce DLs, knowledge bases, and CREs in more detail. We then present the algorithm for computing answers to generalized instance queries and prove its properties in Section 3. In Section 4, we show the results of our empirical evaluation, followed by a discussion of related work in Section 5 and conclusions in Section 6.

2 Preliminaries

Before we can describe our algorithm, we first have to consider some relevant definitions concerning both DLs and CREs.

2.1 Description Logics

The syntax of the DL $\mathcal{ALCIO}$ is defined using a vocabulary consisting of countably infinite disjoint sets N_C of *atomic concepts*, N_R of *atomic roles*, and N_I of *individuals*. A *role* is either an atomic or an *inverse role* $r^-, r \in N_R$.

An $\mathcal{ALCIO}$ concept is defined as

$$C ::= \top \mid \bot \mid A \mid \{o\} \mid \neg C \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid$$
$$\forall R.C \mid \exists R.C,$$

where $A \in N_C$, $o \in N_I$, and R is a role. The concepts $\exists R.C$ and $\forall R.C$ are called *existential* and *universal restriction*, respectively. The DL $\mathcal{ALC}$ is obtained from $\mathcal{ALCIO}$ by disallowing the use of nominals ($\{o\}$) and inverse roles. In the remainder, we use a, b for individuals, A, B for atomic and C, D for (possibly) complex concepts, and r for an atomic and R for a (possibly) non-atomic role.

A *TBox* is a set of concept inclusion axioms $C \sqsubseteq D$. An *ABox* is a set of (concept and role) assertions of the form $C(a)$ and $R(a, b)$, respectively. A *knowledge base* (or *ontology*) $\mathcal{K} = \mathcal{T} \cup \mathcal{A}$ consists of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$. We use $C \equiv D$ to abbreviate $C \sqsubseteq D$ and $D \sqsubseteq C$. With $\mathsf{cons}(\mathcal{K})$, $\mathsf{rols}(\mathcal{K})$, and $\mathsf{inds}(\mathcal{K})$, we denote the sets of atomic concepts, atomic roles, and individuals occurring in $\mathcal{K}$, respectively.

An *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a non-empty set $\Delta^{\mathcal{I}}$, the *domain* of $\mathcal{I}$, and an *interpretation function* $\cdot^{\mathcal{I}}$, that assigns to each $A \in N_C$ a subset $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, to each $r \in N_R$ a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, and to each $a \in N_I$ an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. This assignment is extended to inverse roles by $(r^-)^{\mathcal{I}} = \{\langle e, d \rangle \mid \langle d, e \rangle \in r^{\mathcal{I}}\}$ and, inductively, to complex concepts as $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$, $\bot^{\mathcal{I}} = \emptyset$, $\{o\}^{\mathcal{I}} = \{o^{\mathcal{I}}\}$, $(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$, $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$, $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$, $(\forall r.C)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \langle d, e \rangle \in r^{\mathcal{I}} \to e \in C^{\mathcal{I}}\}$ and $(\exists r.C)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} : e \in C^{\mathcal{I}} : \langle d, e \rangle \in r^{\mathcal{I}}\}$.

An interpretation $\mathcal{I}$ *satisfies* an axiom α, written $\mathcal{I} \models \alpha$, if $\alpha = C \sqsubseteq D$ and $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, $\alpha = C(a)$ and $a^{\mathcal{I}} \in C^{\mathcal{I}}$, or $\alpha = R(a, b)$ and $\langle a^{\mathcal{I}}, b^{\mathcal{I}} \rangle \in R^{\mathcal{I}}$; $\mathcal{I}$ is a *model* of a knowledge base $\mathcal{K}$, written $\mathcal{I} \models \mathcal{K}$, if $\mathcal{I} \models \alpha$ for each $\alpha \in \mathcal{K}$; $\mathcal{K}$ *entails* an axiom α, written $\mathcal{K} \models \alpha$, if every model of $\mathcal{K}$ satisfies α.

For a knowledge base $\mathcal{K} = \mathcal{T} \cup \mathcal{A}$, we say that $\mathcal{K}$ is *(concept) materialized* if $\mathcal{K} \models A(a)$ implies $A(a) \in \mathcal{K}$ for each $A \in \mathsf{cons}(\mathcal{K}) \cup \{\top\}$ and $a \in \mathsf{inds}(\mathcal{K})$; $\mathcal{A}$ is *reduced* if there is no $\mathcal{A}' \subset \mathcal{A}$ such that $\mathcal{A}' \cup \{B(a) \mid A(a) \in \mathcal{A} \text{ and } \mathcal{K} \models A \sqsubseteq B, A, B \in \mathsf{cons}(\mathcal{K})\}$ is materialized, i.e., a reduced ABox is materialized with the most specific atomic concepts.

An $\mathcal{ALC}$ knowledge base $\mathcal{K} = \mathcal{T} \cup \mathcal{A}$ is *Horn* and in *normalized* form if $C \in N_C$ for every $C(a) \in \mathcal{K}$, and every $C \sqsubseteq D \in \mathcal{K}$ is in one of the forms $\top \sqsubseteq A, A \sqsubseteq B, A \sqsubseteq \bot, A \sqcap B \sqsubseteq A', \exists r.A \sqsubseteq B, A \sqsubseteq \exists r.B, A \sqsubseteq \forall r.B$, where $A, A', B \in N_C$ and $r \in N_R$. We use $\mathcal{K}_\forall$ to denote $\{\forall r.B \mid A \sqsubseteq \forall r.B \in \mathcal{T}\}$. W.l.o.g., we assume in the remainder that Horn $\mathcal{ALC}$ knowledge bases are normalized by applying a structural transformation (see e.g. [Kazakov, 2009]) and that the ABox is reduced.

Each consistent Horn $\mathcal{ALC}$ knowledge base $\mathcal{K}$ has a so-called *universal model* $\mathcal{U}$ which is minimal and unique, and contains all the implied facts. In particular, the universal model has a tree-like form of role-connected (anonymous) individuals with named individuals as roots where individuals

are only made equal if it is necessarily entailed by the knowledge base. For query answering, it suffices to consider this universal model [Toman and Weddell, 2019].

2.2 Concept Referring Expressions

Principally, referring expressions serve the purpose of uniquely identifying an object in a certain context through its properties, that also include relations to other objects [Krahmer and van Deemter, 2012]. In DLs, this can be realized by a conjunction of concepts with relations modeled as existential restrictions over inverse roles, which relate an anonymous individual to a named one (expressed as nominal):

Definition 1. *Let $\mathcal{K}$ be a normalized ontology, $C = A_1 \sqcap \ldots \sqcap A_n$ with $A_i \in \mathsf{cons}(\mathcal{K})$, $r \in \mathsf{rols}(\mathcal{K})$, and $a \in \mathsf{inds}(\mathcal{K})$, then a* concept referring expression E *is defined as*

$$E ::= \{a\} \qquad E ::= C \sqcap \exists r^-.(E)$$

and we call a the base individual *of E.*

The reason for describing referring expressions this way lies in the tree model property of DLs, which allows us to identify an anonymous object in terms of a named individual (the base individual) and the (inverse) role path that connects the former to the latter [Toman and Weddell, 2019]. Consider the example taken from Toman and Weddell [2019]:

Example 1. *Let $\mathcal{K} = \mathcal{T} \cup \mathcal{A}$ consist of the TBox $\mathcal{T} = \{A \sqsubseteq \exists r.C, A \sqsubseteq \exists r.D, A \sqsubseteq \forall r.B\}$ and the ABox $\mathcal{A} = \{A(a)\}$, with the three (tree) models of $\mathcal{K}$:*
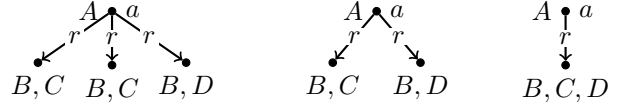
The model in the middle is the universal model as it can be embedded into any other model of $\mathcal{K}$.

When we consider the generalized instance query for the concept B, the universal model makes it reasonable to consider the r-successor of a labeled C and the r-successor of a labeled D as singular certain answers to the query. We can describe these answers as the CREs $C \sqcap \exists r^-.\{a\}$ and $D \sqcap \exists r^-.\{a\}$, respectively. Note that we use an $\mathcal{ALCIO}$ concept to describe the CRE, whereas $\mathcal{K}$ is in Horn $\mathcal{ALC}$.

For a CRE to qualify as an answer to a generalized instance query, we impose some requirements, following the definition of Toman and Weddell [2019] for singularity and certainty:

Definition 2. *A* generalized instance query *is an atomic concept $Q \in N_C$. A set $\mathsf{ans}(Q)$ of CREs is an* answer *to Q over a consistent knowledge base $\mathcal{K}$ if, for each $E, E' \in \mathsf{ans}(Q)$,*

certainty $\mathcal{K} \models E \sqsubseteq Q$ *and $|E^{\mathcal{I}}| > 0$ for all models $\mathcal{I}$ of $\mathcal{K}$,*

singularity $|E^{\mathcal{U}}| = 1$ *in the universal model $\mathcal{U}$ of $\mathcal{K}$, and*

uniqueness $E^{\mathcal{U}} \neq E'^{\mathcal{U}}$ *in the universal model $\mathcal{U}$ of $\mathcal{K}$.*

Here, the additional uniqueness property ensures that we do not obtain any duplicate answers, since we are only interested in the individual (semantically) represented by some CRE rather than the actual (syntactic) form of the latter. Besides, note that the above definition of singularity is a weakening of the property defined by Borgida *et al.* [2016], where a CRE was required to denote a singleton set in *all* models of

the knowledge base. This weakening, however, is essential in DLs that do not support counting (e.g., through number restrictions or functional roles) such as $\mathcal{ALCIO}$. Note also that unreachable objects cannot be certain answers. Furthermore, a complex concept may still be considered as query if it can be represented by means of a new atomic concept and additional appropriate axioms in the ontology. For instance, the query $\top$ may be realized by asking for some new concept A and adding $\top \sqsubseteq A$ to the ontology.

Example 2 (Ex. 1 cont.). *Consider again the knowledge base $\mathcal{K}$ of Example 1. Formally, the set $\{C \sqcap \exists r^-.\{a\}, D \sqcap \exists r^-.\{a\}\}$ is an answer to the generalized instance query B as both concept expressions are singular certain answers that identify different anonymous individuals. The CRE $\exists r^-.\{a\}$ does not satisfy the singularity requirement and the CRE $C \sqcap D \sqcap \exists r^-.\{a\}$ violates the certainty requirement.*

Note that if we were to add $C \equiv D$ to $\mathcal{T}$, the universal model would be the right-most model of Example 1. In addition to the two CREs from above, also $C \sqcap D \sqcap \exists r^-.\{a\}$ becomes a certain singular answer. All three CREs, however, identify the same anonymous individual in the universal model (in different syntactic variants) and our proposed uniqueness criterion requires that $\mathsf{ans}(Q)$ consists of only one of them.

3 Answering Generalized Instance Queries

We propose an algorithm for computing CREs for a generalized instance query given by an atomic concept Q over a (normalized) Horn $\mathcal{ALC}$ ontology $\mathcal{K} = \mathcal{T} \cup \mathcal{A}$ with reduced ABox $\mathcal{A}$. An important aspect of the algorithm is that it employs an OWL reasoner as a black box as only standard reasoning tasks (e.g., as foreseen in the OWL API [Horridge and Bechhofer, 2011]) are required and the concrete approach being applied to answer generalized instance queries is not known to the reasoner. For example, we retrieve super-concepts of a concept or the most specific concepts (w.r.t. to the subsumption hierarchy) that an individual is an instance of (corresponding to those concepts in a reduced ABox). This provides great flexibility concerning the choice of the applied reasoner and allows for using optimized strategies for different ontologies.

To construct CREs serving as answers for Q, we start with a CRE $E = \{a\}$ for each $a \in \mathsf{inds}(\mathcal{K})$ and a corresponding *current concept* $C = A_1 \sqcap \ldots \sqcap A_n$ such that, for each $1 \le i \le n$, $A_i(a) \in \mathcal{A}$. We then look for suitable existential restrictions $\exists r.D$ which satisfy $\mathcal{K} \models C \sqsubseteq \exists r.D$ and, thus, allow us to extend the current CRE to $E' = D \sqcap \exists r^-.(E)$. This procedure is repeated and each time the newly related D is chosen as next current concept until no more appropriate restrictions can be found to further extend the generated CREs. Here, a current concept C does not just enable us to further expand a CRE, but moreover serves as an indicator if the latter represents an answer for the query Q, which is the case if $\mathcal{K} \models C \sqsubseteq Q$ as shown later in Lemma 1.

3.1 Initialization

Algorithm 1 describes the main algorithm that constructs, for each individual a occurring in the knowledge base, the initial

---

Algorithm 1 Answering generalized instance queries

procedure GENERALIZEDINSTANCEQUERY($\mathcal{K}, Q$)
  input: $\mathcal{K} = \mathcal{T} \cup \mathcal{A}$: an ontology, Q: a generalized instance query
  output: $\mathsf{ans}(Q)$
1: $\mathsf{ans} \leftarrow \emptyset$
2: **for all** $a \in \mathsf{inds}(\mathcal{K})$ **do**
3:    $C_a \leftarrow \bigsqcap_{A(a) \in \mathcal{A}} A$
4:    $\mathsf{ans} \leftarrow \mathsf{ans} \cup \text{GETCRES}(\mathcal{K}, Q, C_a, \{a\}, \emptyset)$
5: **return** ans

---

current concept and then calls the algorithm to generate CREs w.r.t. this (base) individual.

3.2 Constructing Concept Referring Expressions

Algorithm 2 uses five steps to construct CREs for a given base individual that serve as certain, singular and unique answers for the considered query:

Step 1: At first, in Line 2, we look for suitable existential restrictions $\exists r.B$ that allow us to relate the current concept C to the concept B in order to establish a link to a new anonymous individual, thus requiring $\mathcal{K} \models C \sqsubseteq \exists r.B$. Since we are only interested in producing singular CREs, a restriction $\exists r.B$ should only be considered if there is no restriction $\exists r.A$ (using the same role r) such that $\mathcal{K} \models A \sqsubseteq B$. We capture this by introducing reduced (sets of) existential restrictions:

Definition 3 (Reduced Existential Restrictions). *Let $\mathcal{K}$ be a normalized Horn $\mathcal{ALC}$ ontology and C a concept, then $\mathcal{K}_\exists(C) = \{\exists r.B \mid \mathcal{K} \models C \sqsubseteq \exists r.B, B \in N_C, r \in N_R\}$. We denote with $\mathcal{K}_\exists^{\min}(C)$ a smallest subset of $\mathcal{K}_\exists(C)$ such that, for each $\exists r.B \in \mathcal{K}_\exists(C) \setminus \mathcal{K}_\exists^{\min}(C)$, there is some $\exists r.A \in \mathcal{K}_\exists^{\min}(C)$ s.t. $\mathcal{K} \models A \sqsubseteq B$.*

Note that an instance of $\exists r.B \in \mathcal{K}_\exists(C) \setminus \mathcal{K}_\exists^{\min}(C)$ is connected to more than one instance of B, which means that a CRE using this existential restriction is non-singular as it represents more than one individual. Moreover, if there are two candidates $\exists r.A, \exists r.B \in \mathcal{K}_\exists(C)$ such that $\mathcal{K} \models A \equiv B$, only one of them should be chosen, because applying both may lead to CREs that describe the same anonymous individual and, hence, represent unnecessary duplicate answers.

Step 2: For each selected existential restriction that induces a successor for an instance of the current concept C, we next consider universal restrictions that further specify these successors (see Lines 4–5) and we add these concepts to the set of *successor concepts* for the role in the sc relation. After this step, each $(r, S) \in \mathsf{sc}$ is such that $\mathcal{K} \models C \sqsubseteq \exists r.(B_1 \sqcap \ldots \sqcap B_n)$ for $S = \{B_1, \ldots, B_n\}$.

Step 3: In this step of the algorithm, we determine those existential restrictions that can actually be applied to further construct CREs leading to *singular*, *unique* answers of the query and collect them in the set next. If we encounter an entry $(r, S) \in \mathsf{sc}$ for which we already have another entry $(r, S') \in \mathsf{next}$ such that $\mathcal{K} \models \bigsqcap_{B \in S} B \equiv \bigsqcap_{B' \in S'} B'$, we do not consider (r, S) any further to prevent duplicate answers, i.e., if the check in Line 11 fails, the entry (r, S) is not added to the set next. Even though we already consider only the reduced set of existential restrictions in Step 1, this check is still

Algorithm 2 Computing CREs for a base individual

---

procedure GETCRES($\mathcal{K}, Q, C, E$, used)
    input: $\mathcal{K} = \mathcal{T} \cup \mathcal{A}$: an ontology, Q: a query, C: the current concept, E: the current CRE, used: a set of already applied existential restrictions
    output: CREs from ans(Q) that build upon E
1:   *// Step 1: find suitable existential restrictions*
2:   sc $\leftarrow \{(r, \{B\}) \mid \exists r.B \in \mathcal{K}_{\exists}^{\min}(C)\}$
3:   *// Step 2: find related universal restrictions*
4:   **for all** $(r, S) \in$ sc **do**
5:     $S \leftarrow S \cup \{B \mid \forall r.B \in \mathcal{K}_{\forall} \text{ and } \mathcal{K} \models C \sqsubseteq \forall r.B\}$
6:   *// Step 3: filter found existential restrictions*
7:   next $\leftarrow \emptyset$
8:   **for all** $(r, S) \in$ sc **do**
9:     *// check for potential duplicates*
10:     sc$' \leftarrow \{(r, S') \in$ next $\mid \mathcal{K} \models \bigsqcap_{B \in S} B \equiv \bigsqcap_{B' \in S'} B'\}$
11:     **if** sc$' = \emptyset$ **then**
12:       **if** $E = \{a\}$ **then**
13:         **if** $\{b \mid r(a, b) \in \mathcal{A}, \forall B \in S : \mathcal{K} \models B(b)\} = \emptyset$ **then**
14:           next $\leftarrow$ next $\cup \{(r, S)\}$
15:       **else**
16:         *// look for cycle in current CRE*
17:         **if** $(r, S) \in$ used **then**
18:           $E \leftarrow$ MARKCYCLE(E, r, S)
19:         **else**
20:           next $\leftarrow$ next $\cup \{(r, S)\}$
21:   *// Step 4: check if current CRE is an answer*
22:   ans $\leftarrow \emptyset$
23:   **if** $\mathcal{K} \models C \sqsubseteq Q$ **then**
24:     ans $\leftarrow$ ans $\cup \{E\}$
25:   *// Step 5: extend CRE recursively*
26:   **for all** $(r, S) \in$ next **do**
27:     $C' \leftarrow \bigsqcap_{B \in S} B$
28:     $E' \leftarrow C' \sqcap \exists r^-.(E)$
29:     ans $\leftarrow$ ans $\cup$ GETCRES($\mathcal{K}, Q, C', E'$, used $\cup \{(r, S)\}$)
30:   **return** ans

---

necessary. Consider, for example, that $\mathcal{K} \models C \sqsubseteq \exists r.A$ and $\mathcal{K} \models C \sqsubseteq \exists r.B$ for the current concept C and $\mathcal{K} \not\models A \equiv B$. Hence, we have two entries $(r, \{A\})$ and $(r, \{B\}) \in$ sc. Assume, furthermore, that $\mathcal{K} \models C \sqsubseteq \forall r.A$ and $\mathcal{K} \models C \sqsubseteq \forall r.B$. This means, however, that for both $(r, \{A\}), (r, \{B\}) \in$ sc, we have $\mathcal{K} \models C \sqsubseteq \exists r.(A \sqcap B)$.

To guarantee the uniqueness property, an existential restriction should also be ignored for the initial extension of a CRE only consisting of a nominal $\{a\}$ if the existential restriction is already satisfied through a named individual b. This is handled in Line 13. Due to the tree model property of Horn $\mathcal{ALC}$ this situation can only occur for the initial CREs.

For cyclic knowledge bases, it may be the case that for two, possibly equal, current concepts C and C' that occur during the algorithm processing, the same existential restriction is applicable. This may lead to an arbitrarily long repetition of the same sequence of existential restrictions that originally led from C to C', hence, giving rise to an infinitely large number of possible CREs (of increasing length). Analogous

to blocking in tableau algorithms [Baader *et al.*, 2003], we ensure termination of the algorithm by detecting such cycles and by preventing the reuse of an already processed existential restriction. Instead, we compactly represent such CREs using a *cycle notation* in form of surrounding square brackets "$[...]^+$" to mark the sequence that may occur several times repeatedly (indicated by a call to the procedure MARKCYCLE in Line 18). For an example, consider $\mathcal{K} = \mathcal{T} \cup \mathcal{A}$ with $\mathcal{T} = \{A \sqsubseteq \exists r.A\}$ and $\mathcal{A} = \{A(a)\}$, where $[A \sqcap \exists r^-.(]^+\{a\})$ is an answer for the generalized instance query A and serves as single representative for the infinitely many CREs of the form $A \sqcap \exists r^-.(A \sqcap \exists r^-.(...\{a\})$. Note that cycles only appear on a syntactic level, because semantically every CRE represents a path in the tree-like universal model.

Step 4: In this step, we check if the current CRE already represents an answer for the query Q. For a standard instance query, this would usually require to check $\mathcal{K} \models Q(a)$ for a named individual a, but since we are considering anonymous individuals represented by some CRE E, we want to know whether $\mathcal{K} \models E \sqsubseteq Q$. This is actually realized by checking $\mathcal{K} \models C \sqsubseteq Q$ in Line 23 for the related current concept C based on the following lemma.

Lemma 1. *Given a Horn $\mathcal{ALC}$ ontology $\mathcal{K}$, a generalized instance query Q, a CRE E, and its associated (Horn $\mathcal{ALC}$) current concept C,[1] then $\mathcal{K} \models E \sqsubseteq Q \Leftrightarrow \mathcal{K} \models C \sqsubseteq Q$.*

Proof sketch. ($\Leftarrow$) Per definition, a current concept C of some CRE E always satisfies $\mathcal{K} \models E \sqsubseteq C$.
($\Rightarrow$) $\mathcal{K} \models C \sqsubseteq Q$ requires that C is at least as specific as Q. Therefore, we show that the current concept C is actually the most specific (Horn $\mathcal{ALC}$) concept such that $\mathcal{K} \models E \sqsubseteq C$ and that for any other most specific candidate D with $\mathcal{K} \models E \sqsubseteq D$, we have $\mathcal{K} \models C \equiv D$, which is mainly based on the fact that we only work with reduced existential restrictions (see Line 2 in Algorithm 2, and Definition 3). $\square$

Step 5: Even if an answer has been identified, the processing of the CRE does still proceed, because it may be possible that the continuing construction leads to another answer later on. Therefore, the algorithm is eventually called recursively using appropriately updated arguments, like an extended CRE E' and the new current concept C'.

3.3 Properties

In the following, we discuss why the described algorithm works properly in the way that for a generalized instance query, only certain, singular, and unique answers are produced w.r.t. Definition 2. Additionally, we show that the algorithm returns every such answer and, furthermore, always terminates. Due to space limitations, we mainly provide proof sketches, while more details are given in an online appendix.[2]

Certainty. In Line 23 of Algorithm 2, a CRE E is selected as answer if $\mathcal{K} \models C \sqsubseteq Q$, which implies $\mathcal{K} \models E \sqsubseteq Q$ according to Lemma 1. In addition, a CRE basically describes

---

[1]if $E = \{a\}$, then $C = A_1 \sqcap \ldots \sqcap A_n$ with $A_i(a) \in \mathcal{A}$, else if $E = D \sqcap \exists r^-.(...)$, then $C = D$

[2]https://www.uni-ulm.de/fileadmin/website_uni_ulm/iui.inst.090/Publikationen/2022/IlGl22a_appendix.pdf

a chain of linked individuals where the existence of one individual ensures the presence of the next one being referenced through an existential restriction. Since we start with an explicitly present individual as base for E, there is always at least on element in $E^{\mathcal{I}}$ for each model $\mathcal{I}$.

Singularity. For an initial CRE $E = \{a\}$, we directly get $|E^{\mathcal{U}}| = 1$ for the universal model $\mathcal{U}$. In the more general case $E = C \sqcap \exists r^-.(E')$ with E' as the (recursive) part leading to the base individual, we show that the non-singular situation $|E^{\mathcal{U}}| > 1$ is never achieved: This would require that there is some part $C' \sqcap \exists r'^-.(E'')$ in E where one individual from $E''^{\mathcal{U}}$ has r'-relations to (at least) two different individuals in $C'^{\mathcal{U}}$. The necessary conditions for this are described in Lemma 2 below (with C' as D), but since the algorithm only selects reduced existential restrictions w.r.t. Definition 3, i.e., some $\exists r.B$ is only chosen if there is no candidate $\exists r.A$ with $\mathcal{K} \models A \sqsubseteq B$, this situation never occurs.

Lemma 2. *Let $\mathcal{K}$ be a Horn $\mathcal{ALC}$ ontology, $\mathcal{U}$ the universal model of $\mathcal{K}$, and assume that $\langle c, a \rangle, \langle c, b \rangle \in r^{\mathcal{U}}$ for some role r, and (anonymous) $a, b \in D^{\mathcal{U}}$ for some concept D. We have $a \neq b$ iff there exist some concepts A, B, C such that $c \in C^{\mathcal{U}}$, $\mathcal{K} \models C \sqsubseteq \exists r.A$, $\mathcal{K} \models C \sqsubseteq \exists r.B$, $\mathcal{K} \models A \sqsubseteq D$, $\mathcal{K} \models B \sqsubseteq D$, and $\mathcal{K} \not\models A \equiv B$.*

Proof sketch. This basically follows from the characteristics of the universal model $\mathcal{U}$. $\qquad\square$

Uniqueness. Considering the tree-like universal model $\mathcal{U}$, having two different CREs E, E' with $E^{\mathcal{U}} = E'^{\mathcal{U}}$ means that i) they both describe the same path in the tree using different concepts or ii) one refers to a sub-path of the other leading to the same anonymous individual: i) Because an edge in a tree path relates to some $\exists r.B$, E' might describe the same path as E if it uses some $\exists r.B'$ with $B \sqsubseteq B'$ (including $B \equiv B'$) for the associated edge instead. In Algorithm 2, this is prevented in Lines 2 and 11. ii) Assuming that a and a' are the base individuals of E and E', respectively, there must be some $r(a, a')$ such that E' refers to a sub-path of E. If $a'^{\mathcal{U}} \in B^{\mathcal{U}}$ for some B, some $\exists r.B$ would describe the same edge as $r(a, a')$, given that $a'^{\mathcal{U}} \in (\exists r.B)^{\mathcal{U}}$, which is why such existential restrictions are not chosen (Line 13).

Completeness. It is not required to construct every possible CRE that represents an answer for the stated query, but rather to find one unique, singular CRE for every, possibly anonymous, individual that serves as an answer. For that, we assume there may exist some CRE E describing a correct, singular and unique answer, but which cannot be constructed by the algorithm and prove that this is not possible: Since the considered E is not generated by the algorithm, it must possess a unique subpart that does not occur in any of the constructed answers. As we already consider all individuals during the initialization, this part must refer to some part $C' \sqcap \exists r^-.(C \sqcap \ldots)$, where the existential restriction $\exists r.C'$ was applied for the concept C. By considering all the cases where C is not used as current concept or $\exists r.C'$ is not selected (or constructed), respectively, we can see that either the certainty, singularity or uniqueness property would be violated by regarding E as an answer. The employed cycle notation may be seen as a compromise to implicitly return a possibly

infinite number of CREs in form of just one compact instance in order to ensure the termination of the algorithm.

Termination. As the given ontology always consists of a finite number of axioms and concepts, all the applied search and iteration processes come to a halt. Therefore, nontermination may only be realized by the repeated usage of an existential restriction during a recursive call, which, however, is prevented by the cycle detection in Line 17 of Algorithm 2.

4 Implementation and Evaluation

A prototypical Java implementation of our algorithm is available online[3] and also includes some optimizations. One of them refers to the ordering of existential and universal restrictions by means of separate subsumption hierarchies. This allows us to reduce the number of performed subsumption tests in the way that if, for example, $\mathcal{K} \not\models C \sqsubseteq \exists r.B$, we do not need to further check $\mathcal{K} \models C \sqsubseteq \exists r'.A$ for any $\exists r'.A$ that is below $\exists r.B$ in the hierarchy, i.e., for which $\mathcal{K} \models \exists r'.A \sqsubseteq \exists r.B$. Usually, reasoners are only concerned with computing subsumption hierarchies for atomic concepts, referred to as *classification* (see e.g. [Baader *et al.*, 2003]). Therefore, we introduce a new atomic concept for each existential (or universal) restriction such that the reasoner can (internally) compute the hierarchies for those new atomic concepts, based on which the actual hierarchies for existential (or universal) restrictions can then be derived. Due to the associated additional overhead, we also implemented the (explicit) Enhanced Traversal Method [Baader *et al.*, 1994] to efficiently deal with smaller number of elements.

Another improvement is achieved by combining base individuals into one set if they share an equivalent initial current concept and furthermore possess analogous given role assertions. Since these properties result in the same computations for each individual, we can avoid unnecessary repetitions by calling the algorithm only once for such a combined set and just generate different versions of the constructed CRE for each base individual in the end.

For a general performance assessment, the implementation was tested on different ontologies listed in Table 1 with codinteraction-A and the separate ore_ont_2608/4516/3313 being part of the ORE 2015 Reasoner Competition Corpus [Matentzoglu and Parsia, 2015], while HAO (v2021-03-05), VO (v1.1.171) and DTO (v1.1.1) were taken from BioPortal[4]. If necessary, axioms not adhering to Horn $\mathcal{ALC}$ were removed and ontologies merged with their imports to obtain one complete ontology.

Tests showed that subsumption checking has a huge influence on the performance, which is why we considered two different reasoners, namely HermiT[5] (v1.3.8) and JFact[6] (v5.0.3). In general, HermiT achieved better runtime results than JFact, except for the larger ontologies VO and DTO, where JFact was faster in determining the earlier mentioned subsumption hierarchies. Since both VO and DTO, however,

---

[3]https://github.com/M-Illich/Computing-CREs

[4]https://bioportal.bioontology.org/ontologies

[5]http://www.hermit-reasoner.com/

[6]http://jfact.sourceforge.net/

Ontology	#con	#ind	#sets	#∃	#∀	Runtime
codinteraction-A	6	21	13	3	0	71
ore_ont_2608	453	212	212	10	10	265
ore_ont_4516	509	217	217	10	10	185
ore_ont_3313	865	2,070	858	114	0	57,966
HAO	2,548	2,764	687	522	1	37,965
VO	6,828	167	4	1,513	237	460*
DTO	10,075	3	1	7,958	114	738*

Table 1: Overview of adapted ontologies with their number of atomic concepts (#con), individuals (#ind), combined individual sets (#sets), considered existential/universal restrictions (#∃/#∀), and average runtime in ms for $\top$ as query (*no subsumption hierarchies)

possess many existential restrictions but only few individuals (especially in form of combined sets), the performance was actually best if no subsumption hierarchies are computed. Table 1 shows the obtained runtime measurements (average of five runs) for computing all CREs, i.e., using $\top$ as query, with HermiT as reasoner, based on an AMD Ryzen 7 3700X 3.59 GHz processor with 16 GB RAM on Windows 10 (64-Bit).

5 Related Work

Krahmer and van Deemter [2012] provide a survey on referring expressions and their computational generation, given by a historic overview of research developments. Here, the problem of generating referring expressions is defined as finding a description in form of combined properties that uniquely identify a given individual in a certain context, which differs from our work in that we intend to describe unknown, anonymous individuals rather than explicitly named ones.

A related approach that works with DLs is provided by Areces *et al.* [2008] introducing an algorithm that takes as input a model $\mathcal{M}$ and tries to find formulas (concepts) that represent the individuals in $\mathcal{M}$. The basic idea is that, starting with $\top$ as formula for which the interpretation contains every domain element, new formulas are created by successively refining the already given ones by appending new conjuncts to reduce the number of elements in the formulas' interpretations. At first, only atomic concepts are added as conjuncts, before existential restrictions relating to already computed formulas are considered until the formulas' interpretations are singular or no further adaptations are possible.

Unlike in our work, the individuals to be described are already stated and the occurrence of cycles in referring expressions is not considered. However, both approaches make use of relations between individuals, but with the distinction that Areces *et al.* start from the individual a described by the referring expression and go to another one b ($a \rightarrow b$), while we utilize the inverse case where another (present) individual b refers to a, the (anonymous) one being described ($a \leftarrow b$).

The work that comes closest to ours is given by Toman and Weddell [2019] dealing with (generalized) instance retrieval queries on (Horn $\mathcal{ALC}$) ontologies, based on a *tree automaton* for which a state S is given by a set of atomic concepts, while transitions are defined by so-called matching tuples $(S, S_0, ..., S_k)$ stating that some S_i with $0 \leqslant i \leqslant k$ can be reached when in the current state S a certain ex-

istential restriction $\exists r_i.C_i$ is applied. Starting with a set $S_a = \{A \mid A(a) \in \mathcal{A}\}$ for some individual a, a tree can be unfolded using those matching tuples. In order to answer a query B, this tree is traversed beginning with some S_a until a state S_k is reached where $\mathcal{K} \models \bigsqcap_{A \in S_k} A \sqsubseteq B$, for which the sequence $r_1 A_1 ... r_k A_k$ (called certain path) referring to the applied existential restrictions is then transformed into a CRE $A_k \sqcap \exists r_k^-.(... A_1 \sqcap \exists r_1^-.\{a\})$ serving as answer for B.

Like in our algorithm, the tree automaton starts the construction of CREs with a given individual and also takes care of occurring cycles. Furthermore, both approaches only use minimal existential restrictions to ensure singularity, although the definitions for minimality are different: while Toman and Weddell rely on the general subsumption of the whole restriction $\exists r.C$, we consider subsumption for the inner concept C instead w.r.t. Definition 3, because only then singularity can be guaranteed (based on Lemma 2). In this respect, the tree automaton approach does not fulfill the completeness property due to only working with the most specific existential restrictions, even though some other ones might lead to singular CREs, too, as illustrated in Example 3 below. Besides, we are only interested in producing unique CREs, which is not explicitly handled by Toman and Weddell.

Example 3. *Let $\mathcal{K} = \mathcal{T} \cup \mathcal{A}$ consist of $\mathcal{A} = \{A(a)\}$ and $\mathcal{T} = \{A \sqsubseteq \exists r.C, \exists r.C \sqsubseteq B, B \sqsubseteq \exists r.D, C \sqsubseteq Q, D \sqsubseteq Q\}$. For the generalized instance query Q, we get $\mathsf{ans}(Q) = \{C \sqcap \exists r^-.\{a\}, D \sqcap \exists r^-.\{a\}\}$, even though $\exists r.D$ is not minimal w.r.t. subsumption due to $\mathcal{K} \models \exists r.C \sqsubseteq \exists r.D$.*

6 Conclusion

We presented an algorithm that generates concept referring expressions to identify both named and anonymous individuals serving as certain, singular and unique answers for an instance retrieval query on a Horn $\mathcal{ALC}$ ontology. For that, we start with a named individual and recursively determine appropriate existential restrictions to create a chain of relations that defines an implicitly given individual based on its connection to an explicit one. A crucial factor here is the selection based on reduced existential restrictions (cf. Definition 3), enabling us to construct every desired singular CRE. Besides, the black box consideration of the applied reasoner provides more flexibility in choosing optimal reasoning approaches for different situations.

Future Research. In general, the question which reasoner should be selected for the considered ontologies is not an easy one and requires further investigations. On the other side, there may be some advantages of directly integrating the algorithm into a particular reasoner as the algorithm can then utilize the reasoner's internal data structures and be optimized accordingly. Moreover, it may be of interest to consider extensions of Horn $\mathcal{ALC}$, too. While some features might be already supported with none or only small adaptations in the algorithm, others probably require more examination, especially since we might lose the existence of the tree-like universal model and thus can no longer make use of our original singularity definition.

References

[Areces *et al.*, 2008] Carlos Areces, Alexander Koller, and Kristina Striegnitz. Referring Expressions as Formulas of Description Logic. In *Proceedings of the Fifth International Natural Language Generation Conference*, pages 42–49, 2008.

[Baader *et al.*, 1994] Franz Baader, Bernhard Hollunder, Bernhard Nebel, Hans-Jürgen Profitlich, and Enrico Franconi. An Empirical Analysis of Optimization Techniques for Terminological Representation Systems. *Applied Intelligence*, 4(2):109–132, 05 1994.

[Baader *et al.*, 2003] Franz Baader, Diego Calvanese, Deborah McGuinness, Peter Patel-Schneider, Daniele Nardi, et al. *The Description Logic Handbook: Theory, implementation and applications*. Cambridge University Press, 2003.

[Borgida *et al.*, 2016] Alexander Borgida, David Toman, and Grant Weddell. On Referring Expressions in Query Answering over First Order Knowledge Bases. In *Proceedings of the 15th International Conference on Principles of Knowledge Representation and Reasoning (KR'16)*, pages 319–328. AAAI Press, 2016.

[Hitzler *et al.*, 2009] Pascal Hitzler, Markus Krötzsch, Bijan Parsia, Peter F. Patel-Schneider, and Sebastian Rudolph, editors. *OWL 2 Web Ontology Language: Primer*. W3C Recommendation, 2009. Available at http://www.w3.org/TR/owl2-primer/.

[Horridge and Bechhofer, 2011] Matthew Horridge and Sean Bechhofer. The OWL API: A Java API for OWL ontologies. *Semant. Web*, 2(1):11–21, Jan 2011.

[Kazakov, 2009] Yevgeny Kazakov. Consequence-Driven Reasoning for Horn $\mathcal{SHIQ}$ Ontologies. In *Proc. of the 21st Int. Joint Conf. on Artificial Intelligence, IJCAI 2009*, pages 2040–2045, 2009.

[Krahmer and van Deemter, 2012] Emiel Krahmer and Kees van Deemter. Computational Generation of Referring Expressions: A Survey. *Computational Linguistics*, 38(1):173–218, 2012.

[Matentzoglu and Parsia, 2015] Nicolas Matentzoglu and Bijan Parsia. ORE 2015 Reasoner Competition Corpus, June 2015.

[Toman and Weddell, 2019] David Toman and Grant Weddell. Finding ALL Answers to OBDA Queries Using Referring Expressions. In *AI 2019: Advances in Artificial Intelligence*, pages 117–129, Cham, 2019. Springer International Publishing.