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: $\text{Person} \sqsubseteq \exists \text{hasMother}.\text{Person}$ and $\text{Person}(\text{john})$, where $\text{Person}$ is a concept, $\text{hasMother}$ is a role, and $\text{john}$ is an individual). Obviously, $\text{john}$ is an answer to the query for instances of the concept $\text{Person}$ 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: $\text{Person} \sqcap \exists \text{hasMother}^{-}.\{\text{john}\}$, where $\text{hasMother}^{-}$ denotes the inverse relation is mother of of $\text{hasMother}$). 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 ALCITO 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 ALCITO concept is defined as

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

where $A \in N_C$, $a \in N_I$, and $R$ is a role. The concepts $\exists R.C$ and $\forall R.C$ are called existential and universal restriction, respectively. The DL ALC is obtained from ALCITO by disallowing the use of nominals $\{a\}$ 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 $A(a)$ and $R(a, b)$, respectively. A knowledge base (ontology) $K = T \cup A$ consists of a TBox $T$ and an ABox $A$. We use $C = D$ to abbreviate $C \sqsubseteq D$ and $D \sqsubseteq C$. With $\text{cons}(K)$, $\text{rols}(K)$, and $\text{inds}(K)$, we denote the sets of atomic concepts, atomic roles, and individuals occurring in $K$, respectively.

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

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

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

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

Each consistent Horn ALC knowledge base $K$ has a so-called universal model $U_K$ 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 [Krämer 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 $K$ be a normalized ontology, $C = A_1 \sqcup \ldots \sqcup A_n$ with $A_i \in \text{cons}(K)$, $r \in \text{rols}(K)$, and $a \in \text{inds}(K)$, then a concept referring expression $E$ is defined as $E ::= \{a\}$ 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 $K = T \cup A$ consist of the TBox $T = \{A \sqsubseteq \exists r.C, A \sqsubseteq \forall r.B\}$ and the ABox $A = \{A(a)\}$, with the three (tree) models of $K$:

```
A   a
  ↙   ↙
B, C B, C B, D
```

The model in the middle is the universal model as it can be embedded into any other model of $K$.

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 $\exists C \sqsubseteq \exists r^{-}.\{a\}$ and $D \sqsubseteq \forall r^{-}.\{a\}$, respectively. Note that we use an ALCITO concept to describe the CRE, whereas $K$ is in Horn 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 $\text{ans}(Q)$ of CREs is an answer to $Q$ over a consistent knowledge base $K$ if, for each $E, E' \in \text{ans}(Q)$,

- **certainty** $K \models E \sqsubseteq Q$ and $|E^I| > 0$ for all models $I$ of $K$,
- **singularity** $|E^I| = 1$ in the universal model $U_K$ of $K$, and
- **uniqueness** $E^I \neq E'^I$ in the universal model $U_K$ of $K$.

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 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 $T$ may be realized by asking for some new concept $A$ and adding $T \subseteq A$ to the ontology.

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

Note that if we were to add $C \equiv D$ to $T$, the universal model would be the right-most model of Example 1. In addition to the two CREs from above, also $C \cap D \cap \exists r^-, \{a\}$ 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 $\text{ans}(Q)$ 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 ALC ontology $K = \mathcal{T} \cup \mathcal{A}$ 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 expression as shown later in Lemma 1. In addition, the two CREs from above, also $C \cap D \cap \exists r^-, \{a\}$ 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 $\text{ans}(Q)$ consists of only one of them.

To construct CREs serving as answers for $Q$, we start with a CRE $E = \{a\}$ for each $a \in \text{inds}(K)$ and a corresponding current concept $C = A_1 \sqcap \ldots \sqcap A_n$ such that, for each $1 \leq i \leq n, A_i(a) \in A$. We then look for suitable existential restrictions $\exists r \cdot D$ which satisfy $K \models C \sqsubseteq \exists r \cdot D$ and, thus, allow us to extend the current CRE to $E' = D \sqcap \exists r^- \cdot (E)$. 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 $K \models C \sqsubseteq Q$ 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 current concept and then calls the algorithm to generate CREs w.r.t. this (base) individual.

3.2 Constructing Concept Referring Expressions

Algorithm 1 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 $\exists r \cdot B$ 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 $K \models C \sqsubset \exists r \cdot B$. Since we are only interested in producing singular CREs, a restriction $\exists r \cdot B$ should only be considered if there is no restriction $\exists r \cdot A$ (using the same role $r$) such that $K \models A \sqsubset B$. We capture this by introducing reduced (sets of) existential restrictions:

Definition 3 (Reduced Existential Restrictions). Let $K$ be a normalized Horn ALC ontology and $C$ a concept, then $K_2(C) = \{\exists r \cdot B \mid K \models C \sqsubset \exists r \cdot B, B \in N_C, r \in N_R\}$. We denote with $K_2^{\text{min}}(C)$ a smallest subset of $K_2(C)$ such that, for each $\exists r \cdot B \in K_2^{\text{min}}(C)$, there is some $\exists r \cdot A \in K_2^{\text{min}}(C)$ s.t. $K \models A \sqsubset B$.

Note that an instance of $\exists r \cdot B \in K_2^{\text{min}}(C)$ 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 $\exists r \cdot A, \exists r \cdot B \in K_2^{\text{min}}(C)$ such that $K \models A \sqsubset B$, 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 $\text{sc}$ relation. After this step, each $(r, S) \in \text{sc}$ is such that $K \models C \sqsubset \exists r \cdot \{B_1 \sqcap \ldots \sqcap B_n\}$ 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 $\text{next set}$. If we encounter an entry $(r, S) \in \text{sc}$ for which we already have another entry $(r, S') \in \text{next set}$ such that $K \models \bigwedge_{B \in S} B \equiv \bigwedge_{B' \in S'} B'$, 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 $\text{next set}$. 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(K, Q, C, E, used)
    input: K = T ∪ 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 ← \{(r, B)\} | \exists r.B ∈ K^{fin}(C)\}
3: // Step 2: find related universal restrictions
4: for all (r, S) ∈ sc do
5:     S ← S \cup \{B | \forall r.B ∈ K_v and K ⊩ C ⇔ \forall r.B\}
6: // Step 3: filter found existential restrictions
7:     next ← ∅
8: for all (r, S) ∈ sc do
9:     // check for potential duplicates
10:     sc′ ← \{(r, S′) ∈ next | K ⊩ B ∈ S ′ ≡ B′\}
11: if sc′ = ∅ then
12:     if E = \{a\} then
13:         if \{b | \exists r.a, \forall B ∈ S : K ⊩ B(b)\} = ∅ then
14:             next ← next \cup \{r, S\}
15: else
16:     // look for cycle in current CRE
17:     if (r, S) ∈ used then
18:         E ← MARKCYCLE(E, r, S)
19:     else
20:         next ← next \cup \{r, S\}
21: // Step 4: check if current CRE is an answer
22:     ans ← ∅
23: if K ⊩ C ⊩ Q then
24:     ans ← ans \cup \{E\}
25: // Step 5: extend CRE recursively
26: for all (r, S) ∈ next do
27:     C′ ← \bigcap_{B ∈ S} B
28: E′ ← C′ \cap \exists r−.(E)
29:     ans ← ans \cup GETCREs(K, Q, C′, E′, used \cup \{(r, S)\})
30: return ans

necessary. Consider, for example, that K ⊩ C ⊩ \exists r.A and K ⊩ C ⊩ \exists r.B for the current concept C and K ⊭ A ⊩ B. Hence, we have two entries (r, \{A\}) and (r, \{B\}) ∈ sc. Assume, furthermore, that K ⊩ C ⊩ \forall r.A and K ⊩ C ⊩ \forall r.B. This means, however, that for both (r, \{A\}), (r, \{B\}) ∈ sc, we have K ⊩ C ⊩ \exists r.(A \cap 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 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 yield 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 repeated (indicated by a call to the procedure MARKCYCLE in Line 18). For an example, consider K ⊩ T ∪ A with T = \{A ⊩ \exists r.A\} and A = \{A(a)\}, where \{A ⊩ \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 ⊩ \exists r−.\{A ⊩ \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 K ⊩ Q(a) for a named individual a, but since we are considering anonymous individuals represented by some CRE E, we want to know whether K ⊩ E ⊩ Q. This is actually realized by checking K ⊩ C ⊩ Q in Line 23 for the related current concept C based on the following lemma.

Lemma 1. Given a Horn ALC ontology K, a generalized instance query Q, a CRE E, and its associated (Horn ALC) current concept C,1 then K ⊩ E ⊩ Q ⇔ K ⊩ C ⊩ Q.

Proof sketch. (⇒) Per definition, a current concept C of some CRE E always satisfies K ⊩ E ⊩ C.
(⇐) K ⊩ C ⊩ 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 ALC) concept such that K ⊩ E ⊩ C and that for any other most specific candidate D with K ⊩ E ⊩ D, we have K ⊩ C ⊩ 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).

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 K ⊩ C ⊩ Q, which implies K ⊩ E ⊩ Q according to Lemma 1. In addition, a CRE basically describes

¹if E = \{a\}, then C = A₁ ∩ \ldots ∩ A_n with A_i(a) ∈ A, else if E = D ∩ \exists r−.\{a\}, then C = D
²https://www.uni-ulm.de/fileadmin/website_uni_ulm/iui.inst.090/Publikationen/2022/IJGI22a_appendix.pdf
a chain of linked individuals where the existence of one individ-
ual ensures the presence of the next one being referenced
through an existential restriction. Since we start with an ex-
plicitly present individual as base for $E$, there is always
at least on element in $E^d$ for each model $T$.

**Singularity.** For an initial CRE $E = \{a\}$, we directly get
$|E^d| = 1$ for the universal model $U$. In the more general case
$E = C \cap \exists r \cdot (E')$ with $E'$ as the (recursive) part lead-
ing to the base individual, we show that the non-singular situa-
tion $|E^d| > 1$ is never achieved: This would require that
there is some part $C' \cap \exists r' \cdot (E'')$ in $E$ where one individual
from $E'^d$ has $r'$-relations to (at least) two different individu-
als in $C'^d$. 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
$K \models A \sqsubseteq B$, this situation never occurs.

**Lemma 2.** Let $K$ be a Horn ALC ontology, $U$ the univer-
sal model of $K$, and assume that $(c, a), (c, b) \in \mu^d$ for
some role $r$, and (anonymous) $a, b \in D^d$ for some concept $D$. We have
$a \neq b$ if there exist some concepts $A, B$ such that $c \in C^d$,
$K \models C \sqsubseteq \exists r.A, K \models C \sqsubseteq \exists r.B, K \models A \sqsubseteq D, K \models B \sqsubseteq D$, and $K \models A \sqsubseteq B$.

**Proof sketch.** This basically follows from the characteristics
of the universal model $U$.

**Uniqueness.** Considering the tree-like universal model $U$,
having two different CREs $E, E'$ with $E^d = E'^d$ means that
i) they both describe the same path in the tree using different
classes 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
$a'^d \in B'$ for some $B$, $\exists r.B$ would describe the same
dge as $r(a, a')$, given that $a'^d \in (\exists r.B)'$, which is why such
existential restrictions are not chosen (Line 13).

**Completeness.** It is not required to construct every possi-
ble 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 individu-
als during the initialization, this part must refer to some part
$C' \cap \exists r \cdot (C \cap \ldots)$, 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 cer-
tainty, 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, non-
termination 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 avail-
able online\(^3\) and also includes some optimizations. One of
them refers to the ordering of existential and universal restric-
tions by means of separate subsumption hierarchies. This
allows us to reduce the number of performed subsumption
tests in the way that if, for example, $K \models C \sqsubseteq \exists r.B,
we do not need to further check $K \models C \sqsubseteq \exists r'.A$ for any
$\exists r'.A$ that is below $\exists r.B$ in the hierarchy, i.e., for which
$K \models \exists r'.A \sqsubseteq \exists r.B$. 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 exist-
ential (or universal) restriction such that the reasoner can (in-
ternally) compute the hierarchies for those new atomic con-
cepts, 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 (ex-
plicit) Enhanced Traversal Method [Baader et al., 1994] to
efficiently deal with smaller number of elements.

Another improvement is achieved by combining base individ-
uals into one set if they share an equivalent initial current
concept and furthermore possess analogous given role asser-
tions. 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 implementa-
tion 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 BioPor-
tal\(^4\). If necessary, axioms not adhering to Horn ALC
were removed and ontologies merged with their imports to
obtain one complete ontology.

Tests showed that subsumption checking has a huge in-
fluence 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/
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).

<table>
<thead>
<tr>
<th>Ontology</th>
<th>#con</th>
<th>#ind</th>
<th>#sets</th>
<th>#∃</th>
<th>#∀</th>
<th>Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>codinteraction-A</td>
<td>6</td>
<td>21</td>
<td>13</td>
<td>3</td>
<td>0</td>
<td>71</td>
</tr>
<tr>
<td>ore_ont_2608</td>
<td>509</td>
<td>217</td>
<td>217</td>
<td>10</td>
<td>0</td>
<td>185</td>
</tr>
<tr>
<td>ore_ont_4516</td>
<td>865</td>
<td>2,070</td>
<td>858</td>
<td>114</td>
<td>0</td>
<td>57,966</td>
</tr>
<tr>
<td>HAO</td>
<td>2,548</td>
<td>2,764</td>
<td>687</td>
<td>522</td>
<td>1</td>
<td>37,965</td>
</tr>
<tr>
<td>VO</td>
<td>6,828</td>
<td>167</td>
<td>4,153</td>
<td>237</td>
<td></td>
<td>460*</td>
</tr>
<tr>
<td>DTO</td>
<td>10,075</td>
<td>3</td>
<td>1</td>
<td>7,958</td>
<td>114</td>
<td>738*</td>
</tr>
</tbody>
</table>

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")

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 $M$ and tries to find formulas (concepts) that represent the individuals in $M$. 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 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.

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

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


