



Commonality Subtraction Operator for the EL Description Logic

Axel Mascaro, Christophe Rey

► To cite this version:

Axel Mascaro, Christophe Rey. Commonality Subtraction Operator for the EL Description Logic. Description Logics 2023, Oliver Kutz, Carsten Lutz, Ana Ozaki, Sep 2023, Rhodes, Greece. hal-04488651

HAL Id: hal-04488651

<https://hal.science/hal-04488651>

Submitted on 4 Mar 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Commonality Subtraction Operator for the \mathcal{EL} Description Logic

Axel Mascaro¹, Christophe Rey¹

¹Université Clermont Auvergne, CNRS, Ecole des Mines de Saint-Etienne, LIMOS, F-63000 Clermont-Ferrand, France.

Abstract

In the context of the \mathcal{EL} description logic, we define and study a new concept difference operator, called commonality subtraction operator (CSO), with respect to an acyclic definitional ontology \mathcal{T} , and noted $A \ominus_{\mathcal{T}} B$. CSO aims at removing from a concept description A all common parts with another description B , w.r.t. \mathcal{T} , which we call descriptonal commonalities. Based on the proposed operator of tree subtraction (TSO), we give an algorithm to compute CSO along with its complexity. CSO fits well with existential restrictions and applies to any couple of concepts (A, B) , which makes it different from existing difference operators. We practically justify the definition of CSO by explaining our needs for such an operator in the context of a metrology resources management project.


Keywords


difference, subtraction, EL, descriptonal commonalities, TSO, CSO

1. Introduction


The STAM project¹, funded by the European Regional Development Fund (FEDER) of the European Union, aims at developing a multitool platform in the field of metrology. One of its objectives is to provide a kind of facebook for metrology. In that purpose, it is based on a documentation repository in which metrological resources (e.g. pdf documents, images, texts, data files, instruments...) could be easily retrieved. In its current version, metrological resources are identified by characteristics defined in a metrology dictionary and which are retrieved by a keyword-based search. Besides, resources are also tagged by annotations called "families", another kind of keywords. By selecting one or many families, the user can restrict the results of a keyword search to resources annotated by the chosen families.


In [1], we have improved this exact retrieval process by generalizing it into a matchmaking one, which aims at finding the semantically closest resources with respect to the user query, using the \mathcal{EL} description logic (DL). The choice of \mathcal{EL} is linked to the underlying metrology resource management system which is powered by GraphDB, an RDF data management system that allows tractable reasoning in the OWL2 EL profile, based on \mathcal{EL} [2]. First, leveraging the existing dictionary, an \mathcal{EL} ontology is built by associating logical descriptions to metrological keywords and families in order to obtain \mathcal{EL} concept definitions. Then, user queries that describe

 DL 2023: 36th International Workshop on Description Logics, September 2–4, 2023, Rhodes, Greece

 axel.mascaro@uca.fr (A. Mascaro); christophe.rey@uca.fr (C. Rey)

 0000-0003-3581-9449 (C. Rey)

 © 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

¹https://limos.fr/news_project/118

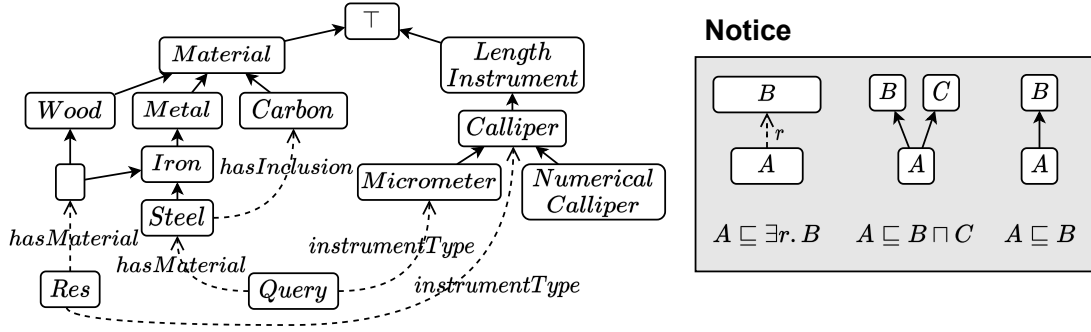


Figure 1: Ontology of the metrology (as an \mathcal{EL} TBox) in example 1.

wanted resources are built as \mathcal{EL} concept descriptions. The semantically closest resources w.r.t. a user query are obtained by pairwise comparing all resources w.r.t the query, in a semantic way using the ontology. This process produces a ranking of resources w.r.t the query based on the idea that the bigger the shared information between the query and the resources, the better. But instead of directly computing the shared information, we compute what is original in each resource w.r.t. the query. This means best resources are the ones which have the least original parts w.r.t. the query. Moreover, the process also computes what parts of the query are original w.r.t. each resource, which can be used to further refine the ranking if needed. The whole approach is based on a new difference operator for \mathcal{EL} , namely the commonality subtraction operator (CSO, noted $\ominus_{\mathcal{T}}$), which is the contribution presented in this paper.

Example 1. In the metrology context, we may have the following \mathcal{EL} ontology and resource and query descriptions (see figure 1): $\mathcal{T} = \{\text{Steel} \sqsubseteq \text{Iron} \sqcap \exists \text{hasInclusion}.\text{Carbon}, \text{Iron} \sqsubseteq \text{Metal}, \text{Metal} \sqsubseteq \text{Material}, \text{Wood} \sqsubseteq \text{Material}, \text{Carbon} \sqsubseteq \text{Material}, \text{Micrometer} \sqsubseteq \text{Calliper}, \text{NumericalCalliper} \sqsubseteq \text{Calliper}, \text{Calliper} \sqsubseteq \text{LengthInstrument}\}$, $\text{Res} = \exists \text{instrumentType}.\text{Calliper} \sqcap \exists \text{hasMaterial}.\text{Iron} \sqcap \text{Wood}$, and $\text{Query} = \exists \text{instrumentType}.\text{Micrometer} \sqcap \exists \text{hasMaterial}.\text{Steel}$. Intuitively, *Steel* is defined as *Iron* in which is included *Carbon*, *Iron* is a kind of *Metal* which is a kind of *Material*, as *Wood* and *Carbon*. *Micrometer* is a kind of *Calliper*, as *NumericalCalliper*, and *Calliper* is a kind of *LengthInstrument*. *Res* describes a resource that is about a *Calliper* as instrument type, made of *Iron* and *Wood*. With *Query*, a user is looking for resources about *Micrometer* as instrument type, made of *Steel*. We then would like to have: (i) $\text{Res} \ominus_{\mathcal{T}} \text{Query} = \exists \text{hasMaterial}.\text{Wood}$, which means *Query* shares all aspects of *Res* except the fact that *Res* is a resource about an instrument made of wood, and (ii) $\text{Query} \ominus_{\mathcal{T}} \text{Res} = \exists \text{instrumentType}.\text{Micrometer} \sqcap \exists \text{hasMaterial}.\exists \text{hasInclusion}.\text{Carbon}$, which means that *Res* shares with *Query* the fact that it describes resources made of *Iron*, but does not share other aspects of *Query* (the *Micrometer* instrument type and the *Carbon* inclusion inside the material).

The CSO ensures two important features: inverse subsumption criterion to define commonalities, and fine-grained difference. The inverse subsumption criterion states that a commonality between the minuend and the subtrahend exists when a part of the minuend subsumes the

Table 1

\mathcal{EL} constructors and axioms. $A \in \mathbf{C}$, $R \in \mathbf{r}$, and C and D are concepts.

Constructors/Axioms	Syntax	Semantics	Remarks
top	\top	$\Delta^{\mathcal{I}}$	It is assumed that: • conjunctions do not contain \top nor many times the same conjunct, and • writing $\prod_{i=1}^n C_i$ means the C_i s are not conjunctions themselves.
concept name $\in \mathbf{C}$	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$	
role $\in \mathbf{r}$	r	$r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$	
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$	
existential restriction	$\exists r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y \in C^{\mathcal{I}} : (x, y) \in r^{\mathcal{I}}\}$	
Concept definition	$A \equiv C$	$A^{\mathcal{I}} = C^{\mathcal{I}}$	A appears only once as the lhs of a definition.
Primitive concept definition	$A \sqsubseteq C$	$A^{\mathcal{I}} \subseteq C^{\mathcal{I}}$	

subtrahend. A contrario, in existing operators, the minuend is usually subsumed by (parts of) the subtrahend. This is justified by our resource retrieval context where the minuend is seen as a query and the subtrahend may answer parts of it: we consider answering part of a query as corresponding to being subsumed by this part of the query. In example 1, considering $Query \ominus_{\mathcal{T}} Res$, $\exists hasMaterial.Iron$ is a commonality between $Query$ and Res since it is a part of $Query$ (once $Steel$ has been replaced by its definition $Iron \sqcap \exists hasInclusion.Carbon$) and it subsumes Res . A contrario, $\exists hasMaterial.Steel$ is not a commonality with Res since it does not subsume Res . The fine-grained difference browses the tree structure implied by existential restrictions in order to precisely remove commonalities between the minuend and the subtrahend without modifying the remaining of the minuend. Going on with the same example, the CSO removes $\exists hasMaterial.Iron$ from $Query$ since it is a commonality with Res , and it keeps $\exists hasMaterial.\exists hasInclusion.Carbon$, and $\exists instrumentType.Micrometer$.

After recalling notions about \mathcal{EL} in section 2, we study CSO for \mathcal{EL} (with an acyclic and definitional TBox) in section 3. In section 4, we relate CSO to other difference operators. At last, we conclude. When not given in the text body, full proofs of properties are given in appendix.

2. Recalls about \mathcal{EL}

We assume to have two countably infinite sets: \mathbf{C} for concept names and \mathbf{r} for role names. From these, with the help of \mathcal{EL} constructors (see table 1), \mathcal{EL} *concept descriptions* can be built. From now on, unless stated otherwise, the term *concept* refers to the expression " \mathcal{EL} concept description". A concept that is not a concept name nor \top is called a *compound concept*. Given a concept C , we can define its size.

Definition 1 (size [3]). *Given a concept C , its size noted $size(C)$ is defined by induction on its structure: if $C \in \mathbf{C} \cup \top$ then $size(C) = 1$; if $C = C_1 \sqcap C_2$, then $size(C) = 1 + size(C_1) + size(C_2)$; and if $C = \exists r.D$ then $size(C) = 1 + size(D)$*

Concepts are given a model-theoretic semantics based on *interpretations* which are couples $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ of, respectively, a universe of discourse and an *interpretation function*, see the third column of table 1. Axioms that relate concepts are of the following kinds: *concept definitions* of

the form $A \equiv C$ and *primitive concept definitions* of the form $A \sqsubseteq C$. The *size* of an axiom is the sum of the sizes of the left and right hand sides of the axiom. An \mathcal{EL} TBox, or just TBox, is a finite set of axioms. The *size* $\text{size}(\mathcal{T})$ of a TBox \mathcal{T} is the sum of the sizes of its axioms. An interpretation $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is a *model* of a TBox \mathcal{T} if, for each axiom in \mathcal{T} , the condition given in the third column of table 1 is satisfied. A concept C is *subsumed by* another concept D w.r.t. a TBox \mathcal{T} , noted $C \sqsubseteq_{\mathcal{T}} D$ (or $\mathcal{T} \models C \sqsubseteq D$), if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ in every model of \mathcal{T} . When $\mathcal{T} = \emptyset$, we can note interchangeably \sqsubseteq or \sqsubseteq_{\emptyset} .

A *definitional TBox* contains only concept definitions. A TBox containing primitive concept definitions can be made definitional in linear time w.r.t. $\text{size}(\mathcal{T})$ [3] since (i) each primitive concept definition $A \sqsubseteq C$ can be transformed into the concept definition $A \equiv C \sqcap \bar{A}$, with \bar{A} a new concept name, and (ii) two primitive concept definitions $A \sqsubseteq B$ and $A \sqsubseteq C$ can be grouped into one $A \sqsubseteq B \sqcap C$. In the sequel, TBoxes are supposed to be definitional.

The *signature* of a TBox \mathcal{T} , noted $\text{sig}_{\mathcal{T}}$, is the set of all concept names and roles that occur in \mathcal{T} . We note $\mathbf{C}_{\mathcal{T}} = \mathbf{C} \cap \text{sig}_{\mathcal{T}}$, $\mathbf{r}_{\mathcal{T}} = \mathbf{r} \cap \text{sig}_{\mathcal{T}}$, and $\mathcal{T}_{\mathcal{EL}}$ the set of all concepts that can be built using elements of $\text{sig}_{\mathcal{T}}$ and \top . Concept names appearing as the left-hand side of a definition are called *defined concepts* and they define the set $\text{def}_{\mathcal{T}} \subseteq \mathbf{C}_{\mathcal{T}}$. Defined concepts may only appear once as the left hand side of a concept definition. Other concept names are called *primitive concepts*. They define the set $\text{prim}_{\mathcal{T}} \subseteq \mathbf{C}_{\mathcal{T}}$. The set of concepts built using only primitive concepts of \mathcal{T} and \top is noted $\mathcal{T}_{\mathcal{EL}}^{\text{prim}}$.

Following definition 2.9 of [3], for A , B and B' concept names, we say that A *directly uses* B in \mathcal{T} if there is in \mathcal{T} a primitive concept definition $A \sqsubseteq C$, or a concept definition $A \equiv C$, such that B occurs in C . We say that A *uses* B if A directly uses B , or if there is a concept name B' such that A uses B' and B' directly uses B . A TBox contains a *cycle* when some concept name A uses itself. A TBox is *acyclic* if it contains no cycle. In the sequel, TBoxes are supposed to be acyclic, in addition to being definitional.

The *complete expansion* (a.k.a. unfolding) \mathcal{T}^* of an acyclic definitional TBox \mathcal{T} [4, 5] rewrites every concept definition of \mathcal{T} into an equivalent one with only primitive concepts in its right hand side. Then, for a concept C , $\mathcal{T}^*(C)$ is the complete expansion of C w.r.t. \mathcal{T} . This process is EXPTIME in the sizes of \mathcal{T} and C .

Example 2. We have the following acyclic definitional TBox: $\mathcal{T} = \{A \equiv B \sqcap C, C \equiv D \sqcap \exists r.B, D \equiv E, F \equiv \exists r.D\}$. Thus we have $\mathbf{C}_{\mathcal{T}} = \{A, B, C, D, E, F\}$, $\mathbf{r}_{\mathcal{T}} = \{r\}$, $\text{prim}_{\mathcal{T}} = \{B, E\}$ and $\text{def}_{\mathcal{T}} = \{A, C, D, F\}$. The complete expansion \mathcal{T}^* is $\mathcal{T}^* = \{A \equiv B \sqcap E \sqcap \exists r.B, C \equiv E \sqcap \exists r.B, D \equiv E, F \equiv \exists r.E\}$.

3. The Commonality Subtraction Operator (CSO)

In section 3.1, we define the CSO, first informally, by presenting the notions of characteristic branch and descriptive commonality, and then formally. In section 3.2, we present a new syntactical operator called the tree subtraction operator (TSO) and show how to use it to compute the CSO. We also give the main properties associated to both the TSO and the CSO, namely existence, unicity, and termination, soundness and complexity of the associated algorithms.

3.1. Definition of CSO

The CSO operator $C \ominus_{\mathcal{T}} D$ is intended to remove from the minuend C all concept parts shared with the subtrahend D w.r.t. some TBox \mathcal{T} . We call these shared parts \mathcal{T} *descriptive commonalities* (or \mathcal{T} commonalities for short) from C to D . Syntactically, we want commonalities to be removable from the minuend without impacting its other parts. So they have to be atomic in some sense. Since \mathcal{EL} concepts have a tree structure (see [6]), we capture the notion of atomic parts of a concept as its *branches* in its tree structure. We define the notion of branch with the ones of *subdescription* and *width* of a concept. Semantically, being a \mathcal{T} commonality from C to D means being a part of C linked to D : we propose a \mathcal{T} commonality from C to D to be defined as a *characteristic branch* of C w.r.t. \mathcal{T} that subsumes D :

- A characteristic branch of C w.r.t. \mathcal{T} is a primitive branch of C or of a concept equivalent to C (w.r.t. \mathcal{T}) such that it cannot be syntactically removed from C without changing its semantics. Fine-grained difference (cf. introduction) is achieved by working at the level of characteristic branches.
- Imposing D being subsumed by a characteristic branch of C expresses the fact that commonalities from C to D are parts of C to which D answers (by being subsumed by them). This is how the inverse subsumption criterion is implemented (cf. introduction).

Then, $C \ominus_{\mathcal{T}} D$ is defined as the minimal concept E that subsumes C such that there are no \mathcal{T} commonalities from E to D (meaning all \mathcal{T} commonalities from C to D have been removed).

We now formalize these notions. First, the width of a concept C is the maximum number of conjuncts composing any conjunction occurring in C .

Definition 2 (width of a concept). *The width of C , noted $\text{wid}(C)$, is defined as follows:*

$\text{wid}(C) = 1$ if $C \in \mathbf{C} \cup \{\top\}$

$\text{wid}(C) = \text{Max}(n, \text{Max}\{\text{wid}(C_i), 1 \leq i \leq n\})$ if $C = \prod_{i=1}^n C_i, n \geq 2$

$\text{wid}(C) = \text{wid}(D)$ if $C = \exists r.D$

A subdescription of a concept C is obtained by removing zero or many conjuncts anywhere in C , provided it remains a syntactically correct concept². The following definition formalizes this idea. It is equivalent to the definition given in [7] (restricted to \mathcal{EL}).

Definition 3 (subdescription of a concept). *With $n \geq 2$, the set of subdescriptions of C , noted subd_C , is set to $\{C\}$ if $C \in \mathbf{C} \cup \{\top\}$, or to $\{\exists r.E \mid E \in \text{subd}_D\}$ if $C = \exists r.D$, or to*

$$\bigcup_{\substack{C \subseteq \{C_i \mid 1 \leq i \leq n\} \\ \text{with } |C| = m \neq 0}} \left(\bigcup_{\substack{\langle S_1, \dots, S_m \rangle \in \prod_{C_i \in C} \text{subd}_{C_i}}} \left(\prod_{j=1}^m S_j \right) \right) \text{ if } C = \prod_{i=1}^n C_i.$$

We note $\text{subd}_{C, \mathcal{T}}^{\text{prim}} = \text{subd}_C \cap \mathcal{T}_{\mathcal{EL}}^{\text{prim}}$ the set of subdescriptions of C where concept names are primitive concepts or \top only (w.r.t. \mathcal{T}).

Example 3. Let $D = A \sqcap \exists r.(B \sqcap C)$. We have:

$$\text{subd}_D = \{A \sqcap \exists r.(B \sqcap C), \exists r.(B \sqcap C), A \sqcap \exists r.C, A \sqcap \exists r.B, \exists r.B, \exists r.C, A\}$$

²The notion of subdescription is close to but not the same as the one of *subconcept* defined in [3]. Informally, a subconcept is any conjunction taken in the original concept from which zero or many conjuncts have been removed (keeping at least one).

A branch is essentially a concept having a width of at most 1.

Definition 4 (branch). Let \mathcal{T} be a TBox and $C \in \mathcal{T}_{\mathcal{EL}}$. The set of branches over \mathcal{T} is noted $\text{br}_{\mathcal{T}}$. The set of primitive branches over \mathcal{T} is noted $\text{br}_{\mathcal{T}}^{\text{prim}}$. The set of branches of C is noted br_C . And the set of primitive branches of C is noted $\text{br}_{C,\mathcal{T}}^{\text{prim}}$. These sets are defined as follows:

$$\begin{aligned}\text{br}_{\mathcal{T}} &= \{S \in \mathcal{T}_{\mathcal{EL}} \mid \text{wid}(S) = 1\} \\ \text{br}_{\mathcal{T}}^{\text{prim}} &= \{S \in \mathcal{T}_{\mathcal{EL}}^{\text{prim}} \mid \text{wid}(S) = 1\} \\ \text{br}_C &= \{S \in \text{subd}_C \mid \text{wid}(S) = 1\} \\ \text{br}_{C,\mathcal{T}}^{\text{prim}} &= \{S \in \text{subd}_{C,\mathcal{T}}^{\text{prim}} \mid \text{wid}(S) = 1\}\end{aligned}$$

Example 4. Using TBox \mathcal{T} from example 2, let G be the following concept:

$G = C \sqcap \exists r_1.(\exists r_2.\exists r_3.\top \sqcap D \sqcap \exists r_4.B)$. Then we have:

$\text{br}_G = \{C, \exists r_1.\exists r_2.\exists r_3.\top, \exists r_1.D, \exists r_1.\exists r_4.B\}$ and $\text{br}_{G,\mathcal{T}}^{\text{prim}} = \{\exists r_1.\exists r_2.\exists r_3.\top, \exists r_1.\exists r_4.B\}$.

A characteristic branch of C w.r.t. \mathcal{T} is a primitive branch of C or of a concept equivalent to C such that it cannot be syntactically removed from C without changing its semantics.

Definition 5 (characteristic branch). Let \mathcal{T} be a TBox and $C \in \mathcal{T}_{\mathcal{EL}}$. The set $\text{char}_C^{\mathcal{T}}$ of characteristic branches of C w.r.t. \mathcal{T} is defined as follows:

$$\text{char}_C^{\mathcal{T}} = \{S \in \text{br}_{\mathcal{T}}^{\text{prim}} \mid \exists C' \in \mathcal{T}_{\mathcal{EL}} \text{ such that } (C \equiv_{\mathcal{T}} C' \text{ and } S \in \text{br}_{C'} \text{ and } \forall C'' \in \mathcal{T}_{\mathcal{EL}} (\text{br}_{C''} = \text{br}_{C'} \setminus \{S\}) \rightarrow (C'' \not\equiv_{\mathcal{T}} C))\}$$

\mathcal{T} commonalities from C to D are defined as characteristic branches of C that subsume D .

Definition 6 (descriptive commonality). Let \mathcal{T} be a TBox and $(C, D) \in (\mathcal{T}_{\mathcal{EL}})^2$. The set $\text{dcom}_{C,D}^{\mathcal{T}}$ of \mathcal{T} descriptive commonalities from C to D is defined as follows:

$$\text{dcom}_{C,D}^{\mathcal{T}} = \{S \in \text{char}_C^{\mathcal{T}} \mid D \sqsubseteq_{\mathcal{T}} S\}$$

At last, $C \ominus_{\mathcal{T}} D$ is defined as the minimal concept E (w.r.t. $\sqsubseteq_{\mathcal{T}}$) that subsumes C with no \mathcal{T} commonalities from E to D (unicity is shown in proposition 2). When all characteristic branches are commonalities (and thus must all be removed from C), the result is \top .

Definition 7 (CSO). Let \mathcal{T} be a TBox and $(C, D) \in (\mathcal{T}_{\mathcal{EL}})^2$. The binary operator $\ominus_{\mathcal{T}}$, called commonality subtraction operator (CSO) for \mathcal{T} , is defined as follows:

$$C \ominus_{\mathcal{T}} D = \begin{cases} \text{Min}_{\sqsubseteq_{\mathcal{T}}} \{E \in \mathcal{T}_{\mathcal{EL}} \mid C \sqsubseteq_{\mathcal{T}} E \text{ and } \text{dcom}_{E,D}^{\mathcal{T}} = \emptyset\} & \text{if } \text{char}_C^{\mathcal{T}} \not\subseteq \text{dcom}_{C,D}^{\mathcal{T}} \\ \top & \text{if } \text{char}_C^{\mathcal{T}} \subseteq \text{dcom}_{C,D}^{\mathcal{T}} \end{cases}$$

Example 5. Table 2 shows two examples of CSO, with \mathcal{T} from example 2: $\text{Res} \ominus_{\mathcal{T}} \text{Query}$ and $\text{Query} \ominus_{\mathcal{T}} \text{Res}$ where $\text{Res} = A \sqcap F \sqcap \exists r.\top \sqcap \exists s.\top$ and $\text{Query} = B \sqcap \exists r.D \sqcap \exists s.E$. In both cases, the corresponding sets of characteristic branches and descriptive commonalities are given.

3.2. Computing the CSO using the Tree Subtraction Operator (TSO)

In order to compute $C \ominus_{\mathcal{T}} D$, we propose an approach based on a syntactical difference operator that operates on branches of expansions of C and D . This operator is not the classical set difference of the branch sets since it takes into account subsumption relationships between

Table 2

An example of $Res \ominus_{\mathcal{T}} Query$ and $Query \ominus_{\mathcal{T}} Res$, with \mathcal{T} from example 2.

$\mathcal{T} = \{A \equiv B \sqcap C, C \equiv D \sqcap \exists r.B, D \equiv E, F \equiv \exists r.D\}$	
$Res = A \sqcap F \sqcap \exists r.\top \sqcap \exists s.\top$	$Query = B \sqcap \exists r.D \sqcap \exists s.E$
$\mathcal{T}^*(Res) = B \sqcap E \sqcap \exists r.B \sqcap \exists r.E \sqcap \exists r.\top \sqcap \exists s.\top$	$\mathcal{T}^*(Query) = B \sqcap \exists r.E \sqcap \exists s.E$
$\text{char}_{Res}^{\mathcal{T}} = \{B, E, \exists r.B, \exists r.E, \exists s.\top\}$	$\text{char}_{Query}^{\mathcal{T}} = \{B, \exists r.E, \exists s.E\}$
$\text{dcom}_{Res, Query}^{\mathcal{T}} = \{B, \exists r.E, \exists s.\top\}$ and thus $Res \ominus_{\mathcal{T}} Query = E \sqcap \exists r.B$	
$\text{dcom}_{Query, Res}^{\mathcal{T}} = \{B, \exists r.E\}$ and thus $Query \ominus_{\mathcal{T}} Res = \exists s.E$	

branches, e.g. those involving the concept \top . Moreover it does not change the original tree structure of the minuend. We call this operator the tree subtraction operator (TSO), noted Δ . Informally, $C \Delta D$, to be read " C deprived of D ", is intended to be the minimal concept w.r.t. \sqsubseteq that subsumes C such that all branches in br_C that subsume a branch in br_D have been removed from br_C . That means we remove from br_C branches that are also in br_D , but also branches of br_C that end by \top when there is in br_D a branch beginning with the same existential restrictions. If all branches of C are removed, then the result is set to be \top .

Definition 8 (TSO). *Let C and D be two concepts. The binary operator Δ , called tree subtraction operator (TSO), is defined as follows:*

$$C \Delta D = \begin{cases} \text{Min}_{\sqsubseteq} \{E \mid C \sqsubseteq E \text{ and } \text{br}_E = \text{br}\} & \text{if } \text{br} \neq \emptyset \\ \top & \text{if } \text{br} = \emptyset \end{cases}$$

with $\text{br} = \text{br}_C \setminus (\text{br}_D \cup \{S = \exists r_1.\exists r_2...\exists r_n.\top \in \text{br}_C, n \geq 0 \mid$
 $\exists S' = \exists r_1...\exists r_n.\exists r_{n+1}...\exists r_{n+m}.P \in \text{br}_D, m \geq 0\})$

and P any concept name or \top (with the exception that P cannot be \top when $m = 0$).

Example 6. *Following table 2, with $R = \mathcal{T}^*(Res)$ and $Q = \mathcal{T}^*(Query)$, we have:*

$$\text{br}_R \setminus (\text{br}_Q \cup \{S = \exists r_1.\exists r_2...\exists r_n.\top \in \text{br}_R, n \geq 0 \mid$$

 $\exists S' = \exists r_1...\exists r_n.\exists r_{n+1}...\exists r_{n+m}.P \in \text{br}_Q, m \geq 0\}) = \{E, \exists r.B\}.$

Thus $Res \ominus_{\mathcal{T}} Query = R \Delta Q = E \sqcap \exists r.B$.

It is not difficult to show that definition 8 ensures $C \Delta D$ keeps the tree structure of C , i.e. is a subdescription of C . We can illustrate this with non equivalent concepts having the same set of branches, like $C_1 = \exists r.A \sqcap \exists r.B$ and $C_2 = \exists r.(A \sqcap B)$ which set of branches is $\text{br}_{C_1} = \text{br}_{C_2} = \{\exists r.A, \exists r.B\}$. Suppose we remove from C_1 (resp. C_2) a concept D that has no commonality with C_1 (resp. C_2), then the result is C_1 and not C_2 (resp. C_2 and not C_1), thus keeping the initial tree structure.

TSO is implemented by algorithm 1. Its principle is to traverse at the same time the tree structures of both C and D , removing from C branches that subsume, w.r.t. the empty TBox, a branch of D . Properties of the TSO and algorithm 1 (unicity, termination, soundness, complexity) are given in proposition 1.

Proposition 1 (Properties of TSO and algorithm 1). *Let C and D be two concepts. We have:*

- $C \Delta D$ always exists and is unique.
- Algorithm 1 terminates and produces a unique result.

Algorithm 1 $\text{tso}(C, D)$

Require: C and D two \mathcal{EL} concepts.

Ensure: $C \Delta D$ (cf. def. 8)

```
1: if  $C = D$  or  $C = \top$  then
2:    $Result := \top$ 
3: else
4:   if  $C = C_1 \sqcap \dots \sqcap C_n$  with  $n \geq 2$  then
5:      $Result1 := \text{tso}(C_1, D) \sqcap \dots \sqcap \text{tso}(C_n, D)$ 
6:     if There is at least one conjunct  $\neq \top$  in  $Result1$  then
7:        $Result := Result1$  without any  $\top$  conjunct.
8:     else
9:        $Result := \top$ 
10:    end if
11:  else if  $D = D_1 \sqcap \dots \sqcap D_m$  with  $m \geq 2$  then
12:     $Result := \text{tso}(\dots (\text{tso}(\text{tso}(C, D_1), D_2), \dots), D_m)$ 
13:  else if  $C = \exists r.C'$  and  $D = \exists r.D'$  then
14:     $Result1 := \text{tso}(C', D')$ 
15:    if  $Result1 = \top$  then
16:       $Result := \top$ 
17:    else
18:       $Result := \exists r.Result1$ 
19:    end if
20:  else
21:     $Result := C$ 
22:  end if
23: end if
24: return  $Result$ 
```

Algorithm 2 $\text{cso}(\mathcal{T}, C, D)$

Require:

- \mathcal{T} an acyclic and definitional \mathcal{EL} TBox
- $(C, D) \in (\mathcal{T}_{\mathcal{EL}})^2$

Ensure: $C \ominus_{\mathcal{T}} D$ (cf. def. 7)

```
1: return  $\text{tso}(\mathcal{T}^*(C), \mathcal{T}^*(D))$ 
```

c. $C \Delta D = \text{tso}(C, D)$ (soundness).

d. Computing $\text{tso}(C, D)$ is in PTIME in the sizes of C and D .

Sketch of proof. a. Existence comes from the definition of br which always corresponds to at least one concept, or \top when it is empty. Unicity trivially comes from the minimality w.r.t. \sqsubseteq .

b. We show by induction on the sizes of C and D that $\text{tso}(C, D)$ always terminates and generates an output which size is strictly less than $\text{size}(C) + \text{size}(D)$.

c. Soundness is showed in 3 steps: (i) from the characterization of subumption in \mathcal{EL} without

Table 3

Comparison of existing difference operators on an example where $Res_1 \equiv A \sqcap B \sqcap \exists r.(C \sqcap D)$ and $Res_2 \equiv A \sqcap B \sqcap \exists r.C$.

Oper.	$Res_1 - Res_2$	$Res_2 - Res_1$
\ominus_{Te}	$\exists r.(C \sqcap D)$	undefined
\ominus_{Br}	$\exists r.(C \sqcap D)$	\top
\ominus_{Su}	\top	$\exists r.C$
\ominus_{He}	$B \sqcap \exists r.(C \sqcap D)$	Res_2
\ominus_{Ri}	Either $\exists r.D$ (full meet case) or one of $\{B \sqcap \exists r.(C \sqcap D), A \sqcap \exists r.(C \sqcap D), A \sqcap B \sqcap \exists r.D\}$ (maxi-choice case)	Res_2 (full meet and maxi-choice cases)
$\ominus_{\mathcal{T}}$	$\exists r.D$	\top

any TBox given in [6], we derive a characterization of subsumption in \mathcal{EL} in terms of subdescriptions, (ii) then is derived a characterization of TSO in terms of subdescriptions, and (iii) at last a proof by induction of soundness is given, using the characterization obtained at step (ii).

d. Tractability is showed by finding a worst case (when C and D are conjunctions of concepts names, without any existential restriction) and studying the complexity in this case. \square

Now, we can use TSO to compute CSO: $C \ominus_{\mathcal{T}} D$ is obtained by computing $\mathcal{T}^*(C) \Delta \mathcal{T}^*(D)$, cf. algorithm 2. Proposition 2 shows properties associated to CSO, namely existence, uniqueness, and soundness and complexity of algorithm 2 (termination is trivially implied by terminations of the complete expansion and algorithm 1).

Proposition 2. *Let \mathcal{T} be a TBox and $(C, D) \in (\mathcal{EL})^2$. There is:*

- a. $C \ominus_{\mathcal{T}} D$ exists and is unique (up to $\equiv_{\mathcal{T}}$).
- b. $C \ominus_{\mathcal{T}} D = \mathcal{T}^*(C) \Delta \mathcal{T}^*(D) = cso(\mathcal{T}, C, D)$ (soundness).
- c. Computing $cso(\mathcal{T}, C, D)$ is in EXPTIME in the sizes of \mathcal{T} , C and D and in PTIME in the sizes of $\mathcal{T}^*(C)$ and $\mathcal{T}^*(D)$.

Sketch of proof. a. Existence and unicity of CSO easily come from definition 7.

b. Soundness of $cso(\mathcal{T}, C, D)$ is grounded on (i) the characterization of subsumption in terms of subdescriptions already used in proof of proposition 1, and on (ii) a lemma stating $br_{C \Delta D}$ is the set of branches of C that do not subsume D .

c. The result easily comes from complexity of the complete expansion and algorithm 1. \square

4. Related works

As far as we know, five difference operators have been defined for DLs before CSO, in [8, 7, 9, 10, 11]. In the sequel, we respectively note them \ominus_{Te} , \ominus_{Br} , \ominus_{Su} , \ominus_{He} and \ominus_{Ri} (and $\ominus_{\mathcal{T}}$ for CSO). We do not consider the contraction operator defined in [12], since it appears to be more a matchmaking operator rather than a difference one.

We first begin by illustrating how these difference operators behave. Let's take the following two descriptions: $Res_1 \equiv A \sqcap B \sqcap \exists r.(C \sqcap D)$ and $Res_2 \equiv A \sqcap B \sqcap \exists r.C$. In table 3, we give

Table 4

Comparison of difference operators in DLs (precise definitions of operators are recalled in appendix).

Reference and informal principle	①	②	③	④	Remarks and complexity
[8] $C \ominus_{Te} D$ finds the maximal concept w.r.t. \sqsubseteq that added to D by conjunction gives a concept equivalent to C .	C	M	M	N	<ul style="list-style-type: none"> • Defined for any DL \mathcal{L}. • Not defined if $C \not\sqsubseteq D$. • Unstudied with a TBox. • Complexity is not given.
[7] $C \ominus_{Br} D$ finds the minimal concepts w.r.t. the subdescription order such that, added to D by conjunction, they give a concept equivalent to $C \sqcap D$.	C	M	T	N	<ul style="list-style-type: none"> • C is in \mathcal{ALC}, D is in \mathcal{ALE}. • Unstudied with a TBox. • PTIME in sizes of C and D given an oracle for subsumption.
[9] $C \ominus_{Su} D$ removes the minimal conjuncts of C , w.r.t. a syntactical total order on \mathcal{EL} concepts, that are subsumed by conjuncts of D (one removed conjunct for one conjunct of D).	S	B	—	N	<ul style="list-style-type: none"> • Defined for \mathcal{EL}. • \mathcal{T} is acyclic and definitional • EXPTIME in the sizes of \mathcal{T}, C and D and PTIME in the sizes of $\mathcal{T}^*(C)$ and $\mathcal{T}^*(D)$.
[10] $C \ominus_{He} D$ removes conjuncts of C that are subsumed by the smallest conjunct of D , w.r.t. a syntactical total order. Recursive process inside existential restrictions.	S	B	—	Y	<ul style="list-style-type: none"> • Defined for \mathcal{EL}. • \mathcal{T} is acyclic and definitional. • $C \ominus_{He} D = C$ if $C \not\sqsubseteq D$. • Complexity is not given.
[11] Extending the notion of subdescription to contain concepts obtained after replacing a concept name by \top , $C \ominus_{Ri} D$ finds the single subdescription (full meet mode) or the many subdescriptions (maxi-choice mode) S of C that are minimal w.r.t. \sqsubseteq such that $S \sqsubseteq D$.	S	M	—	Y	<ul style="list-style-type: none"> • Defined for \mathcal{EL}. • \mathcal{T} is acyclic and definitional. • $C \ominus_{Ri} D = C$ if $C \not\sqsubseteq D$. • Complexity is not given.
[This paper] $C \ominus_{\mathcal{T}} D$ removes \mathcal{T} descriptonal commonalities from C to D , i.e. the characteristic branches of C that subsume D .	S	B	M	Y	<ul style="list-style-type: none"> • Defined for \mathcal{EL}. • \mathcal{T} is acyclic and definitional. • EXPTIME in the sizes of \mathcal{T}, C and D and PTIME in the sizes of $\mathcal{T}^*(C)$ and $\mathcal{T}^*(D)$.

the result of $Res_1 \ominus Res_2$ and $Res_2 \ominus Res_1$, for \ominus replaced by each operator. Note that we suppose to have an empty TBox \mathcal{T} , for a sake of simplicity. We can see that each of the six operators give different results when considering both $Res_1 \ominus Res_2$ and $Res_2 \ominus Res_1$.

Second, we propose to classify these operators in table 4 according to 4 dimensions. ① is their type: S for subtraction-based (something is removed from the minuend) or C for completion-based (something is added to the subtrahend). ② is the definition type of the difference: M for semantical or B for both semantical and syntactical. ③ is the optimization (min or max) criterion type to choose the best result: T for syntactical, M for semantical, or — if non relevant. ④ is the fine grained difference property i.e. the ability to remove precise subdescriptions of the minuend inside nested existential restrictions without removing unnecessary ones: Y for

yes and N for no.

Tables 3 and 4 show that CSO is original w.r.t. existing operators. More precisely:

- \ominus_{Te} and \ominus_{Br} are completion-based operators, they do not achieve fine-grained difference, and \ominus_{Br} does not ensure unicity.
- \ominus_{Su} is a subtraction operation, however it is not a fine-grained difference. Moreover the notion of commonality is based on subsumption and not inverse subsumption.
- \ominus_{He} and \ominus_{Ri} are a fine-grained subtraction operators, as is CSO. However, the differences with CSO are the following ones: (i) for both operators, there can be subtraction only when the minuend is subsumed by the subtrahend, which is a restriction CSO does not have; (ii) in \ominus_{He} , the notion of common part is not based on inverse subsumption; and (iii) \ominus_{Ri} is sometimes too much fine-grained, which may lead it not to remove existential restrictions in cases where CSO does, e.g. $\exists r.A \ominus_{Ri} \exists r.A = \exists r.\top$, while $\exists r.A \ominus_{\mathcal{T}} \exists r.A = \top$.

5. Conclusion

We propose a difference operator for \mathcal{EL} w.r.t. an acyclic and definitional TBox, named CSO. It is based on the TSO, a operator to achieve a syntactical tree difference between two concepts. We propose a tractable algorithm to compute TSO and thus CSO (in the sizes of the complete expansion of the inputs w.r.t. the TBox), and show that CSO is an original difference operator w.r.t existing ones. Also, an implementation of these operators has been done in the Ruby programming language for integration into the metrology platform. Performance tests are being achieved.

Even if CSO is intended to be used in a matchmaking process, we plan to study how it instantiates properties that are defined for difference operators in the AGM approach of agent belief revision [13], namely preservation, success, inclusion, vacuity, recovery, failure, fullness and relevance. This would provide a more precise insight on CSO w.r.t. existing operators and help decide its potential interest in the AGM framework. Besides, we would like to extend this work to the case where TBoxes can be general and cyclic, for which we do not know any DL difference operator yet.

Acknowledgements

This work has been fully supported by the European Regional Development Fund³.

References

- [1] A. Mascaro, C. Rey, Recommandation ontologique multicritère pour la métrologie, in: Rencontres des Jeunes Chercheur·ses en Intelligence Artificielle, in PFIA 2020, Angers, 2020.

³https://ec.europa.eu/regional_policy/en/funding/erdf/.

- [2] D. Calvanese, G. D. Giacomo, J. Carroll, I. Herman, B. Parsia, P. F. Patel-Schneider, A. Ruttenberg, U. Sattler, M. Schneider, Owl 2 web ontology language profiles (second edition), <https://www.w3.org/TR/owl2-profiles/>, 2012. Accessed: 2023-06-14.
- [3] F. Baader, I. Horrocks, C. Lutz, U. Sattler, An Introduction to Description Logic, Cambridge University Press, 2017. URL: <http://www.cambridge.org/de/academic/subjects/computer-science/knowledge-management-databases-and-data-mining/introduction-description-logic?format=PB#17zVGeWD2TZUeu6s.97>.
- [4] B. Nebel, Terminological reasoning is inherently intractable, *Artificial Intelligence* 43 (1990) 235–249.
- [5] The Description Logic Handbook: Theory, Implementation and Applications, 2 ed., Cambridge University Press, 2007. doi:10.1017/CBO9780511711787.
- [6] F. Baader, R. Küsters, R. Molitor, Computing least common subsumers in description logics with existential restrictions, in: T. Dean (Ed.), Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, IJCAI 99, Stockholm, Sweden, July 31 - August 6, 1999. 2 Volumes, 1450 pages, Morgan Kaufmann, 1999, pp. 96–103. URL: <http://ijcai.org/Proceedings/99-1/Papers/015.pdf>.
- [7] S. Brandt, R. Küsters, A.-Y. Turhan, Approximation and difference in description logics, in: Proceedings of the Eighth International Conference on Principles of Knowledge Representation and Reasoning (KR'2002), Toulouse, France, D. Fensel, F. Giunchiglia et al., Eds. San Francisco, California, USA: Morgan Kaufmann Publishers, 2002.
- [8] G. Teege, Making the difference: A subtraction operation for description logics, in: J. Doyle, E. Sandewall, P. Torasso (Eds.), Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning (KR'94). Bonn, Germany, May 24-27, 1994, Morgan Kaufmann, 1994, pp. 540–550.
- [9] F. M. Suchanek, C. Menard, B. M., C. C., Can you imagine... a language for combinatorial creativity?, in: First Part of the Proceedings of the 15th International Semantic Web Conference (ISWC'2016), Kobe, Japan, volume 9981 of *Lecture Notes in Computer Science*, P. T. Groth, E. Simperlet al., Eds., Cham, Switzerland:Springer International Publishing, 2016, pp. 532 – 548.
- [10] H. Heinz, How I Lost My OWL: Retracting Knowledge from EL Concepts, Master's thesis, Koblenz-Landau University, 2018.
- [11] T. Rienstra, C. Schon, S. Staab, Concept contraction in the description logic el, in: International Conference on Principles of Knowledge Representation and Reasoning (KR-20), 2020, pp. 723–732.
- [12] S. Colucci, T. Di Noia, E. Di Sciascio, F. Donini, M. Mongiello, Concept abduction and contraction in description logics, in: Proceedings of the 16th International Workshop on Description Logics (DL'03), volume 81 of *CEUR Workshop Proceedings*, 2003.
- [13] C. E. Alchourrón, P. Gärdenfors, D. Makinson, On the logic of theory change: Partial meet contraction and revision functions, *The Journal of Symbolic Logic* 50 (1985) 510–530.

Appendices

.1. Difference operators in DLs

The definitions of difference operators listed in section 4 are recalled in table 5 with their precise definitions.

.2. Proof of Proposition 1

Proposition 1 (Properties of TSO and algorithm 1). *Let C and D be two concepts. We have:*

- a. $C \Delta D$ always exists and is unique.*
- b. Algorithm 1 terminates and produces a unique result.*
- c. $C \Delta D = \text{tso}(C, D)$ (soundness).*
- d. Computing $\text{tso}(C, D)$ is in PTIME in the sizes of C and D .*

Proof.

a. Existence and unicity of $C \Delta D$ (for the case $\text{br} \neq \emptyset$)

We prove $C \Delta D$ always exists. Since br_E is a subset of br_C , it is always possible to build E such that $C \sqsubseteq E$. Then either E is minimal w.r.t. \sqsubseteq , or it can be made smaller w.r.t. \sqsubseteq by replacing conjunctions of the kind $\exists r.A \sqcap \exists r.B$ into $\exists r.(A \sqcap B)$ until this is not possible while ensuring $C \sqsubseteq E$. So $C \Delta D$ always exists.

We now prove unicity of $C \Delta D$ by contradiction. Suppose there are 2 concepts E_1 and E_2 , with $E_1 \neq E_2$ in $C \Delta D$. Since $\text{br}_{E_1} = \text{br}_{E_2}$ and $E_1 \neq E_2$, we have $E_1 \not\sqsubseteq E_2$. Thus the only possible situation is that E_1 and E_2 are not comparable w.r.t. \sqsubseteq . This implies that $E_1 \sqcap E_2 \sqsubseteq E_1$ and $E_1 \sqcap E_2 \sqsubseteq E_2$. But at the same time, we have $\text{br}_{E_1 \sqcap E_2} = \text{br}_{E_1} = \text{br}_{E_2}$ and $C \sqsubseteq E_1 \sqcap E_2$. This means neither E_1 nor E_2 were minimal w.r.t. \sqsubseteq having the same properties. So, $C \Delta D$ is unique.

b. Termination By induction, we show algorithm 1 always terminates and generates an output whose size, we call s_{out} , is strictly less than the combined size of the inputs C and D , we call s_{in} and define as $s_{in} = \text{size}(C) + \text{size}(D)$. The induction is made on s_{in} .

According to definition 1, we have $\text{size}(C) \geq 1$ and $\text{size}(D) \geq 1$.

Base case: $s_{in} = \text{size}(C) + \text{size}(D) = 1 + 1 = 2$

- Lines 1 and 2: the algorithm clearly stops with $s_{out} = \text{size}(\top) = 1 < 2 = s_{in}$
- Lines 4 to 10: this case is not possible when $\text{size}(C) = 1$.
- Lines 11 and 12: this case is not possible when $\text{size}(D) = 1$.
- Lines 13 to 19: this case is not possible when $\text{size}(C) = \text{size}(D) = 1$.
- Lines 20 and 21: the algorithm clearly stops with $s_{out} = \text{size}(C) = 1 < 2 = s_{in}$

General case: the induction hypothesis (IH) says there is $n \geq 3$ such that the algorithm stops with $s_{out} < s_{in}$ for all $s_{in} = \text{size}(C) + \text{size}(D) \leq n$.

We now suppose $s_{in} = \text{size}(C) + \text{size}(D) = n + 1$. So $\text{size}(C) = n + 1 - \text{size}(D)$ and $\text{size}(D) = n + 1 - \text{size}(C)$ (with $\text{size}(C) \geq 1$ and $\text{size}(D) \geq 1$).

Table 5

Comparison of difference operators in DLs, with their formal definition.

Definition	Ref., [①,②,③,④] and remarks
$C \ominus_{Te} D = \max_{\sqsubseteq} \{E \in \mathcal{L} \mid D \sqcap E \equiv C\}$	[8], [C,M,M,N] <ul style="list-style-type: none"> • \mathcal{L} is any DL. • Not defined if $C \not\sqsubseteq D$.
$C \ominus_{Br} D = \min_{\preceq} \{E \in \mathcal{ALC} \mid E \sqcap D \equiv C \sqcap D\}$	[7], [C,M,T,N] <ul style="list-style-type: none"> • \preceq is the subdescription ordering defined for \mathcal{ALC} concepts. • C is in \mathcal{ALC} and D is in \mathcal{ALE}.
Case 1: D is a one conjunct conjunction. $C \ominus_{Su} D = \begin{cases} \text{norm}(\bigcap_{\substack{1 \leq i \leq n, \\ i \neq k}} C_i) & \text{if } \exists k \mid k = \text{argmin}_i \{C_i \mid C_i \sqsubseteq D\} \\ C & \text{otherwise} \end{cases}$ Case 2: D is a conjunction of at least 2 conjuncts. $C \ominus_{Su} D = (\dots((C \ominus_{Su} D_1) \ominus_{Su} \dots \ominus_{Su} D_m)$	[9], [S,B,-,N] <ul style="list-style-type: none"> • Defined in \mathcal{EL}. • $\text{norm}(D) = D_1 \sqcap \dots \sqcap D_m$ • $\text{norm}(C) = C_1 \sqcap \dots \sqcap C_n$ • $\forall 1 \leq i \leq n-1, C_i \prec C_{i+1}$ with \prec a syntactical total order for \mathcal{EL} concepts.
If $C \not\sqsubseteq D$ then $C \ominus_{He} D = C$. Else (i.e. $C \sqsubseteq D$): $C \ominus_{He} D = \begin{cases} \top & \text{if } C \text{ is a primitive concept} \\ \bigcap_{i=1}^n (C_i \ominus_{He} D_{\prec}) & \text{if } C = C_1 \sqcap \dots \sqcap C_n \\ \exists r.(C_1 \ominus_{He} D_1) & \text{if } C = \exists r.C_1, D_{\prec} = \exists r.D_1, \\ & \text{and } C_1 \ominus_{He} D_1 \neq \top \\ \top & \text{if } C = \exists r.C_1, D_{\prec} = \exists r.D_1, \\ & \text{and } C_1 \ominus_{He} D_1 \equiv \top \\ \top & \text{if } D \equiv \top \\ C & \text{otherwise} \end{cases}$	[10], [S,B,-,Y] <ul style="list-style-type: none"> • C_i is not a conjunction, $\forall i$. • D_{\prec} is the least conjunct of D w.r.t. \prec. • \prec a syntactical total order for \mathcal{EL} concepts. • C and D are normalized and completely expanded. • C and D are in \mathcal{EL}.
If $C \not\sqsubseteq D$ then $C \ominus_{Ri} D = C$. Else (i.e. $C \sqsubseteq D$): $C \ominus_{Ri} D \in LCS(\sigma(C \perp D))$ with $C \perp D = \{[E] \mid$ 1. $C \sqsubseteq E$ 2. $E \not\sqsubseteq D$ 3. E minimal w.r.t. \sqsubseteq s.t. 1. and 2. and $[E] = \{E' \mid E' \equiv E\}$.	[11], [S,M,-,Y] <ul style="list-style-type: none"> • σ is a selection function s.t.: if $C \perp D \neq \emptyset$, $\sigma(C \perp D) \neq \emptyset$. if $C \perp D \neq \emptyset$, $\sigma(C \perp D) \subseteq C \perp D$. if $C \perp D = \emptyset$, $\sigma(C \perp D) = \{[C]\}$. • Maxi-choice case: when σ selects a unique concept. Full meet case: when σ selects all concepts. • C and D are in \mathcal{EL}.
$C \ominus_{\mathcal{T}} D = \begin{cases} \text{Min}_{\sqsubseteq_{\mathcal{T}}} \{E \in \mathcal{T}_{\mathcal{EL}} \mid C \sqsubseteq_{\mathcal{T}} E \text{ and } \text{dcom}_{E,D}^{\mathcal{T}} = \emptyset\} & \text{if } \text{char}_{\mathcal{T}}^{\mathcal{T}} \not\subseteq \text{dcom}_{C,D}^{\mathcal{T}} \\ \top & \text{if } \text{char}_{\mathcal{T}}^{\mathcal{T}} \subseteq \text{dcom}_{C,D}^{\mathcal{T}} \end{cases}$ with $\text{dcom}_{C,D}^{\mathcal{T}} = \{S \in \text{char}_{\mathcal{T}}^{\mathcal{T}} \mid D \sqsubseteq_{\mathcal{T}} S\}$ and $\text{char}_{\mathcal{T}}^{\mathcal{T}} = \{S \in \text{br}_{\mathcal{T}}^{\text{prim}} \mid \exists C' \in \mathcal{T}_{\mathcal{EL}} \text{ such that}$ $(C \equiv_{\mathcal{T}} C' \text{ and } S \in \text{br}_{C'} \text{ and}$ $\forall C'' \in \mathcal{T}_{\mathcal{EL}} (\text{br}_{C''} = \text{br}_{C'} \setminus \{S\}) \rightarrow (C'' \not\equiv_{\mathcal{T}} C))\}$	[This paper], [S,B,-,Y] <ul style="list-style-type: none"> • C and D are in \mathcal{EL}.

- Lines 1 and 2: the algorithm clearly stops with $s_{out} = \text{size}(\top) = 1 < n + 1 = s_{in}$ (since $n \geq 3$).
- Lines 4 to 10:
 - Since $C = C_1 \sqcap \dots \sqcap C_k$, with $k \geq 2$, there is (according to definition 1):
 $\text{size}(C) = (\sum_{i=1}^k \text{size}(C_i)) + k - 1$, with $\text{size}(C_i) \geq 1, \forall 1 \leq i \leq k$
 Since $\text{size}(C) = n + 1 - \text{size}(D)$, we have $\forall 1 \leq i \leq k$:
 $\text{size}(C_i) = n + 1 - \text{size}(D) - \text{size}(C_1) - \dots - \text{size}(C_{i-1}) - \text{size}(C_{i+1}) - \dots - \text{size}(C_k) - k + 1$
 As $\forall 1 \leq i \leq k, \text{size}(C_i) \geq 1$, we have:
 $\text{size}(C_i) \leq n + 1 - \text{size}(D) - (k - 1) - k + 1 = n - \text{size}(D) - 2.k + 3$
 - At line 5, we have $k \geq 2$ recursive calls $\text{tso}(C_i, D)$ with $\text{size}(C_i) \leq n - \text{size}(D) - 2.k + 3$ and $\text{size}(D) \geq 1$. For each recursive call $\text{tso}(C_i, D)$, we note the combined size of its inputs s_{in}^{rc} and the size of its output s_{out}^{rc} . Then, for each recursive call $\text{tso}(C_i, D)$, we have:
 - $s_{in}^{rc} = \text{size}(C_i) + \text{size}(D) \leq n - \text{size}(D) - 2.k + 3 + \text{size}(D) = n - 2.k + 3 \leq n - 1 \leq n$ (since $k \geq 2$).
 - Thus, by IH, the recursive call stops and $s_{out}^{rc} < s_{in}^{rc} \leq n - 2.k + 3$, i.e. $s_{out}^{rc} \leq n - 2.k + 2$
 - At line 7, since all recursive calls stop, the algorithm stops with $s_{out} \leq k * (n - 2.k + 2) + k - 1$ (since there are k concepts C_i and $k - 1$ constructors \sqcap in C).
 - Now, $k \leq (n + 1)/2$. Indeed, $\text{size}(C) \leq n$ (since $\text{size}(C) + \text{size}(D) = n + 1$ and $\text{size}(D) \geq 1$). And $\text{size}(C) \geq 2.k - 1$ (since $\text{size}(C) = (\sum_{i=1}^k \text{size}(C_i)) + k - 1$ and $\text{size}(C_i) \geq 1, \forall 1 \leq i \leq k$).
 - Thus $s_{out} \leq (n + 1)/2 * (n - 2.(n + 1)/2 + 2) + (n + 1)/2 - 1 = n$
 So $s_{out} < n + 1 = s_{in}$.
 - At line 9, the algorithm clearly stops with $s_{out} = \text{size}(\top) = 1 < n + 1 = s_{in}$ (since $n \geq 3$).
- Lines 11 and 12:
 - Here we focus on $D = D_1 \sqcap \dots \sqcap D_m$, with $m \geq 2$. The same reasoning as the one made at line 4 with D instead of C leads to
 - $\forall 1 \leq j \leq m$:
 $\text{size}(D_j) = n + 1 - \text{size}(C) - \text{size}(D_1) - \dots - \text{size}(D_{j-1}) - \text{size}(D_{j+1}) - \dots - \text{size}(D_m) - m + 1$,
 - $\forall 1 \leq j \leq m$: $\text{size}(D_j) \leq n - \text{size}(C) - 2.m + 3$ and
 - $m \leq (n + 1)/2$
 - Now we show the set of nested recursive calls
 $\text{tso}(\dots (\text{tso}(\text{tso}(C, D_1), D_2), \dots), D_m)$
 terminates with $s_{out} < s_{in}$.
 Formally this would need a proof by induction. However, for a sake of simplicity, we only sketch this proof. We use the notations $s_{in}^{rc,j}$ and $s_{out}^{rc,j}$ for the combined size of the inputs and the size of the output of the nested recursive call involving D_j as its second argument.

- First the recursive call $\text{tso}(C, D_1)$ is such that $s_{in}^{rc,1} = \text{size}(C) + \text{size}(D_1)$.
Thus: $s_{in}^{rc,1} \leq \text{size}(C) + n - \text{size}(C) - 2.m + 3 = n - 2.m + 3$
Thus: $s_{in}^{rc,1} \leq n - 1 \leq n$ (since $m \geq 2$).
By IH, the recursive call stops with $s_{out}^{rc,1} < s_{in}^{rc,1} \leq n - 2.m + 3$.
- The recursive call $\text{tso}(\text{tso}(C, D_1), D_2)$ is such that
 $s_{in}^{rc,2} = \text{size}(s_{out}^{rc,1}) + \text{size}(D_2)$
And then:
 $s_{in}^{rc,2} < \text{size}(s_{in}^{rc,1}) + \text{size}(D_2)$
 $s_{in}^{rc,2} < \text{size}(s_{in}^{rc,1})$
 $+ n + 1 - \text{size}(C) - \text{size}(D_1) - \text{size}(D_3) - \dots - \text{size}(D_m) - m + 1$
 $s_{in}^{rc,2} \leq \text{size}(s_{in}^{rc,1})$
 $+ n + 1 - \text{size}(C) - \text{size}(D_1) - \text{size}(D_3) - \dots - \text{size}(D_m) - m$
 $s_{in}^{rc,2} \leq \text{size}(C) + \text{size}(D_1)$
 $+ n + 1 - \text{size}(C) - \text{size}(D_1) - \text{size}(D_3) - \dots - \text{size}(D_m) - m$
 $s_{in}^{rc,2} \leq n + 1 - \text{size}(D_3) - \dots - \text{size}(D_m) - m$
 $s_{in}^{rc,2} \leq n + 1 - (m - 2) - m = n - 2.m + 3$ (since $\text{size}(D_j) \geq 1, \forall 1 \leq j \leq m$)
By IH, the recursive call stops with $s_{out}^{rc,2} < s_{in}^{rc,2} \leq n - 2.m + 3$.
- The recursive call $\text{tso}(\text{tso}(\text{tso}(C, D_1), D_2), D_3)$ is such that
 $s_{in}^{rc,3} = \text{size}(s_{out}^{rc,2}) + \text{size}(D_3)$
And then:
 $s_{in}^{rc,3} < \text{size}(s_{in}^{rc,2}) + \text{size}(D_3)$
 $s_{in}^{rc,3} < \text{size}(s_{in}^{rc,2})$
 $+ n + 1 - \text{size}(C) - \text{size}(D_1) - \text{size}(D_2) - \text{size}(D_4) - \dots - \text{size}(D_m) - m + 1$
 $s_{in}^{rc,3} \leq \text{size}(s_{in}^{rc,2})$
 $+ n + 1 - \text{size}(C) - \text{size}(D_1) - \text{size}(D_2) - \text{size}(D_4) - \dots - \text{size}(D_m) - m$
 $s_{in}^{rc,3} \leq \text{size}(s_{out}^{rc,1}) + \text{size}(D_2)$
 $+ n + 1 - \text{size}(C) - \text{size}(D_1) - \text{size}(D_2) - \text{size}(D_4) - \dots - \text{size}(D_m) - m$
 $s_{in}^{rc,3} < \text{size}(s_{in}^{rc,1}) + \text{size}(D_2)$
 $+ n + 1 - \text{size}(C) - \text{size}(D_1) - \text{size}(D_2) - \text{size}(D_4) - \dots - \text{size}(D_m) - m$
 $s_{in}^{rc,3} < \text{size}(C) + \text{size}(D_1) + \text{size}(D_2)$
 $+ n + 1 - \text{size}(C) - \text{size}(D_1) - \text{size}(D_2) - \text{size}(D_4) - \dots - \text{size}(D_m) - m$
 $s_{in}^{rc,3} \leq \text{size}(C) + \text{size}(D_1) + \text{size}(D_2)$
 $+ n + 1 - \text{size}(C) - \text{size}(D_1) - \text{size}(D_2) - \text{size}(D_4) - \dots - \text{size}(D_m) - m - 1$
 $s_{in}^{rc,3} \leq n + 1 - \text{size}(D_4) - \dots - \text{size}(D_m) - m - 1$
 $s_{in}^{rc,3} \leq n + 1 - (m - 2) - m = n - 2.m + 3$ (since $\text{size}(D_j) \geq 1, \forall 1 \leq j \leq m$)
By IH, the recursive call stops with $s_{out}^{rc,3} < s_{in}^{rc,3} \leq n - 2.m + 3$.
- So on and so forth.

As sketched previously, a formal subproof by induction would show that each recursive call in

$\text{tso}(\dots(\text{tso}(\text{tso}(C, D_1), D_2), \dots), D_m)$

stops with $s_{out}^{rc,j} < s_{in}^{rc,j} \leq n - 2.m + 3 < n + 1, \forall 1 \leq j \leq m$. Thus, at line 12, the algorithm stops with $s_{out} < s_{in} = n + 1$.

- Lines 13 to 19:

- Here we have $C = \exists r.C'$ and $D = \exists r.D'$. So $s_{in} = n + 1 = \text{size}(C) + \text{size}(D) \geq 4$.
- At line 14, the recursive call $\text{tso}(C', D')$ is such that $s_{in}^{rc} = \text{size}(C') + \text{size}(D') = n + 1 - 2 = n - 1 \leq n$. By IH, the recursive call stops with $s_{out}^{rc} < s_{in}^{rc} = n - 1$. I.e. $s_{out}^{rc} \leq n - 2$.
- Since instruction at lines 15 to 19 stops, then the algorithm stops. Moreover:
 - At line 16, $s_{out} = \text{size}(\top) = 1 < s_{in}$ (since $s_{in} \geq 4$).
 - At line 18, $s_{out} = 1 + s_{out}^{rc} \leq 1 + n - 2 = n - 1$. So $s_{out} < s_{in} = n + 1$.
- Lines 20 and 21: the algorithm clearly stops with $s_{out} = \text{size}(C) < \text{size}(C) + \text{size}(D) = s_{in}$ (since $\text{size}(D) \geq 1$).

This ends the proof of termination.

Besides, the fact that algorithm 1 generates a unique output comes from the fact that each output is uniquely defined (see lines 2, 7, 9, 12, 16, 18 and 21) and that there is no undeterministic step (i.e. no instruction implying a choice).

c. Soundness The proof of soundness comes in 3 steps:

- Step 1 from the characterization of subsumption in \mathcal{EL} without any TBox given in [6], we derive a characterization of subsumption in \mathcal{EL} in terms of subdescriptions (in corollary 1).
- Step 2 from corollary 1 is derived a characterization of TSO in terms of subdescriptions.
- Step 3 a proof by induction of soundness is given, using the characterization obtained at step 2.

Step 1

Let's recall the characterization of subsumption in \mathcal{EL} w.r.t. empty Tboxes [6]. First let's define what is the description tree of a concept.

Definition 9 (description tree [6]). *Let C be an \mathcal{EL} concept. Its description tree \mathcal{G}_C is a tree represented as a quadruple (V, E, v_0, l) where V is the set of vertices, $E \subseteq V \times \mathbf{r} \times V$ is the edge set, $v_0 \in V$ is the root and $l : V \rightarrow 2^{\mathbf{C}}$ a function that labels vertices with sets of concept names (the empty set stands for \top).*

[6] explains \mathcal{G}_C is unique, how to obtain \mathcal{G}_C from C , or C from \mathcal{G}_C , and that $C_{\mathcal{G}_C} \equiv C$. Along with the definition of an homomorphism between two description trees recalled below, they characterize subsumption of concepts expressed with primitive concepts only.

Definition 10 (homomorphism of description trees [6]). *A homomorphism from a description tree $\mathcal{G}_D = (V_D, E_D, w_0, l_D)$ to a description tree $\mathcal{G}_C = (V_C, E_C, v_0, l_C)$ is a mapping $\varphi : V_D \rightarrow V_C$ such that (1) $\varphi(w_0) = v_0$, (2) $l_D(w) \subseteq l_C(\varphi(w))$, $\forall w \in V_D$, and (3) $(\varphi(w_1), r, \varphi(w_2)) \in E_C, \forall (w_1, r, w_2) \in E_D$.*

Theorem 1 ([6]). *Let C and D be concepts and \mathcal{G}_C and \mathcal{G}_D their corresponding description trees. Then:*

$C \sqsubseteq D$ iff there exists a homomorphism φ from \mathcal{G}_D to \mathcal{G}_C .

Note that, given a TBox \mathcal{T} , theorem 1 also holds with $\sqsubseteq_{\mathcal{T}}$ instead of \sqsubseteq when $(C, D) \in (\mathcal{T}_{\mathcal{EL}}^{\text{prim}})^2$. Now, corollary 1 reformulates theorem 1 in terms of subdescriptions.

Corollary 1. *Let C and D be concepts. Then :*

$C \sqsubseteq D$ iff there exists $D' \in \text{subd}_C$ such that D' can be obtained by applying anywhere in D zero or many times the following syntactic rules (r1) and (r2) that amount to replacing their left hand side by their right hand side:

$$(r1) \exists r.G \sqcap \exists r.H \rightsquigarrow \exists r.(G \sqcap H)$$

$$(r2) \top \rightsquigarrow H$$

with r any role, and G and H any concepts.

Proof. We note:

$$\triangle C \sqsubseteq D$$

$$\square \text{ there exists a homomorphism } \varphi \text{ from } \mathcal{G}_D \text{ to } \mathcal{G}_C.$$

$$\diamond \text{ there exists } D' \in \text{subd}_C \text{ such that } D' \text{ can be obtained by applying anywhere in } D \text{ zero or many times rules (r1) and (r2).}$$

We now show the proof of corollary 1 by showing \diamond implies \triangle and \square implies \diamond .

Proof of \diamond implies \triangle

This implication easily comes from the two following facts:

- $C \sqsubseteq D'$ since $D' \in \text{subd}_C$
- and $D' \sqsubseteq D$ since applying (r1) or (r2) to D clearly results in a more specific concept.

Proof of \square implies \diamond

We prove the result by induction on $\text{size}(D)$.

- Base case: we assume $\text{size}(D) = 1$ (i.e. $D \in \mathbf{C} \cup \{\top\}$). We assume \square . Let's show \diamond .
By definition of $\mathcal{G}_D = (V_D, E_D, w_0, l_D)$, there is: $l_D(w_0) = \{D\}$ if $D \in \mathbf{C}$ or $l_D(w_0) = \emptyset$ if $D = \top$. According to \square and definition 10, we have (with φ a homomorphism from \mathcal{G}_D to \mathcal{G}_C):
 - if $D \in \mathbf{C}$: $l_D(w_0) = \{D\} \subseteq l_C(\varphi(w_0)) = l_C(v_0)$. So $D' = D$ ensures \diamond (i.e. D' can be obtained without applying neither (r1) nor (r2) and $D' \in \text{subd}_C$).
 - if $D = \top$: any subdescription of C can define D' (applying rule (r2) to D) so that \diamond is true.
- General case: we assume $\text{size}(D) = n + 1$ with $n \geq 1$. We assume the induction hypothesis (IH), i.e. for all concepts which size is at most n , there is: \square implies \diamond . So we assume \square for D . Let's show \diamond .
 - Case 1: $D = D_1 \sqcap D_2$ We obviously have $\star \text{size}(D_1) \leq n - 1$ and $\text{size}(D_2) \leq n - 1$. As $D_1 \in \text{subd}_D$ and $D_2 \in \text{subd}_D$ and there exists a homomorphism φ from \mathcal{G}_D to \mathcal{G}_C (since we assume \square for D), then $\star \varphi$ is a homomorphism from \mathcal{G}_{D_1} to \mathcal{G}_C and a homomorphism from \mathcal{G}_{D_2} to \mathcal{G}_C . Thanks to \star and $\star \star$, IH can be applied: there exists $D'_1 \in \text{subd}_C$ (resp. $D'_2 \in \text{subd}_C$) that can be obtained by applying (r1) or (r2) zero or many times anywhere in D . D' can then be built so that its description tree $\mathcal{G}_{D'} = (V_{D'}, E_{D'}, v_0, l_{D'})$ is such that:

$$V_{D'} = V_{D'_1} \cup V_{D'_2},$$

$$E_{D'} = E_{D'_1} \cup E_{D'_2},$$

and $l_{D'} : V_{D'} \rightarrow 2^{\mathbf{C}}$

$$v' \mapsto l_{D'}(v') \text{ with}$$

$$l_{D'}(v') = \begin{cases} l_{D'_1}(v') & \text{if } l_{D'_2}(v') \text{ not defined} \\ l_{D'_2}(v') & \text{if } l_{D'_1}(v') \text{ not defined} \\ l_{D'_1}(v') \cup l_{D'_2}(v') & \text{otherwise} \end{cases}$$

In $\mathcal{G}_{D'}$, we thus have:

$V_{D'} \subseteq V_C$ since $V_{D'_1} \subseteq V_C$ and $V_{D'_2} \subseteq V_C$,

$E_{D'} \subseteq E_C$ since $E_{D'_1} \subseteq E_C$ and $E_{D'_2} \subseteq E_C$,

and $\forall v' \in V_{D'}, l'_{D'}(v') \subseteq l_c(v')$.

So $D' \in \text{subd}_C$. Now the two following arguments show that D' can be obtained by applying (r1) or (r2) zero or many times anywhere in D :

- D'_1 (resp. D'_2) is a subdescription of D' and can be obtained by applying (r1) or (r2) zero or many times anywhere in D_1 (resp. in D_2) which is itself a subdescription of D . Thus, all branches of $\mathcal{G}_{D'}$ in which edges $(v_1, r, v_2) \in E_{D'}$ only come from edges in either $\mathcal{G}_{D'_1}$ or $\mathcal{G}_{D'_2}$ correspond to subdescriptions of D'_1 or D'_2 that can be obtained by applying (r1) or (r2) zero or many times anywhere in D .
 - Now, for all $(v_1, r, v_2) \in E_{D'}$, if (v_1, r, v_2) is both in $E_{D'_1}$ and $E_{D'_2}$, it means that there exists $(w_1, r, w_2) \in E_{D_1}$ and $(w_3, r, w_4) \in E_{D_2}$ (with $w_2 \neq w_4$) such that $(v_1, r, v_2) = (\varphi(w_1), r, \varphi(w_2)) = (\varphi(w_3), r, \varphi(w_4))$. Now, in terms of concepts, this amounts to apply rule (r2) in D to couples of existential restrictions that correspond to (w_1, r, w_2) and (w_3, r, w_4) . Thus, all branches of $\mathcal{G}_{D'}$ in which some edges $(v_1, r, v_2) \in E_{D'}$ come from both $\mathcal{G}_{D'_1}$ and $\mathcal{G}_{D'_2}$ correspond to subdescriptions D_1 or D_2 of D to which (r2) has been applied. As D_1 and D_2 are themselves obtained by applying (r1) or (r2) zero or many times anywhere in D , them so is D' .
- Case 2: $D = \exists r.D_1$.

Clearly, $\star \text{size}(D) = n$ and $D_1 \in \text{subd}_D$.

According to \ominus , there exists a homomorphism φ from \mathcal{G}_D to \mathcal{G}_C . Thus, there exists a concept C' such that $\exists r.C' \in \text{subd}_C$ with $\star \varphi$ being a homomorphism from \mathcal{G}_{D_1} to $\mathcal{G}_{C'}$. Let's assume that v_0 is the root of \mathcal{G}_C and (v_0, r, v_1) is the edge in \mathcal{G}_C starting in v_0 and corresponding to the existential restriction in $\exists r.C'$ (i.e. v_1 is the root of $\mathcal{G}_{C'}$).

Thanks to \star and $\star \varphi$, we can apply IH: there exists $D'_1 \in \text{subd}_{C'}$ that can be obtained by applying (r1) or (r2) zero or many times anywhere in D_1 . D' can then be built so that its description tree $\mathcal{G}_{D'} = (V_{D'}, E_{D'}, v_0, l_{D'})$ is such that:

$V_{D'} = V_{D'_1} \cup \{v_0\}$,

$E_{D'} = E_{D'_1} \cup (v_0, r, v_1)$,

and $l_{D'} : V_{D'} \rightarrow 2^{\mathbf{C}}$

$$v' \mapsto l_{D'}(v') = \begin{cases} l_{D'_1}(v') & \text{if } v' \in V_{C'} \\ l_D(w_0) & \text{if } v' = v_0 = \varphi(w_0) \end{cases}$$

In other words, $D' = \exists r.D'_1$. Then it is easy to see that $D' \in \text{subd}_C$.

Moreover, D' can be obtained by applying (r1) or (r2) zero or many times anywhere in D since D'_1 can be obtained by applying (r1) or (r2) zero or many times anywhere in D_1 , which is a subdescription of D .

□

Step 2

Let's recall the definition of TSO:

$$C \Delta D = \begin{cases} \text{Min}_{\sqsubseteq} \{ \text{concept } E \mid \textcircled{1} C \sqsubseteq E \text{ and } \textcircled{2} \text{br}_E = \text{br} \} & \text{if } \text{br} \neq \emptyset \\ \top & \text{if } \text{br} = \emptyset \end{cases}$$

$$\text{with } \text{br} = \text{br}_C \setminus \left(\text{br}_D \cup \{ S = \exists r_1. \exists r_2 \dots \exists r_n. \top \in \text{br}_C, n \geq 0 \mid \right. \\ \left. \exists S' = \exists r_1 \dots \exists r_n. \exists r_{n+1} \dots \exists r_{n+m}. P \in \text{br}_D, m \geq 0 \} \right)$$

In the case where $\text{br} \neq \emptyset$, ② implies $\text{br}_E \subseteq \text{br}_C$. Now corollary 1 says that: ① $C \sqsubseteq E$ iff there exists $E' \in \text{subd}_C$ such that E' can be obtained by applying anywhere in E zero or many times rules (r1) and (r2), with:

$$(r1) \exists r. G \sqcap \exists r. H \rightsquigarrow \exists r. (G \sqcap H)$$

$$(r2) \top \rightsquigarrow H$$

Since $\text{br}_E \subseteq \text{br}_C$, it means that E' is obtained from E without applying (r2). Now suppose E' has been obtained from E by applying at least once (r1). This means there is in E a subdescription $S_1 = \exists r_1 \dots \exists r_{i-1}. (\exists r_i. G \sqcap \exists r_i. H)$ and there is in C a subdescription $S_2 = \exists r_1 \dots \exists r_i. J$, with J obtained from $G \sqcap H$ by applying zero or many times (r1). It is clear that $S_2 \sqsubseteq S_1$. But this contradicts the fact that E is minimal w.r.t. \sqsubseteq such that ① and ②. So $E' = E$ and thus $E \in \text{subd}_C$. Since $E \in \text{subd}_C$ implies $C \sqsubseteq E$, we have:

$$C \Delta D = \begin{cases} \text{Min}_{\sqsubseteq} \{ E \mid \textcircled{1} E \in \text{subd}_C \text{ and } \textcircled{2} \text{br}_E = \text{br} \} & \text{if } \text{br} \neq \emptyset \\ \top & \text{if } \text{br} = \emptyset \end{cases}$$

$$\text{with } \text{br} = \text{br}_C \setminus \left(\text{br}_D \cup \{ S = \exists r_1. \exists r_2 \dots \exists r_n. \top \in \text{br}_C, n \geq 0 \mid \right. \\ \left. \exists S' = \exists r_1 \dots \exists r_n. \exists r_{n+1} \dots \exists r_{n+m}. P \in \text{br}_D, m \geq 0 \} \right)$$

Since a set of branches determines a unique subdescription for the associated concept, we have:

$$C \Delta D = \begin{cases} E \mid \textcircled{1} E \in \text{subd}_C \text{ and } \textcircled{2} \text{br}_E = \text{br} & \text{if } \text{br} \neq \emptyset \\ \top & \text{if } \text{br} = \emptyset \end{cases}$$

$$\text{with } \text{br} = \text{br}_C \setminus \left(\text{br}_D \cup \{ S = \exists r_1. \exists r_2 \dots \exists r_n. \top \in \text{br}_C, n \geq 0 \mid \right. \\ \left. \exists S' = \exists r_1 \dots \exists r_n. \exists r_{n+1} \dots \exists r_{n+m}. P \in \text{br}_D, m \geq 0 \} \right)$$

Step 3

On the basis of the previous characterization of TSO, we now show $C \Delta D = \text{tso}(C, D)$ by induction on the combined size of the inputs C and D : this combined size is called s_{in} and set up as $\text{size}(C) + \text{size}(D)$.

Base case: $s_{in} = \text{size}(C) + \text{size}(D) = 1 + 1 = 2$

- Lines 1 and 2:

- If $C = D = \top$, $\text{br} = \{\top\} \setminus \{\top\} = \emptyset$ and thus $C \Delta D = \top$, and $\text{tso}(C, D) = \top$ also.
- If $C = \top$ (and $C \neq D$), $\text{br} = \{\top\} \setminus (\text{br}_D \cup \{\top\}) = \emptyset$ and thus $C \Delta D = \top$, and $\text{tso}(C, D) = \top$ also.
- If $C = D$ (and $C \neq \top$), $\text{br} = \text{br}_C \setminus (\text{br}_C \cup \emptyset) = \emptyset$ and thus $C \Delta D = \top$, and $\text{tso}(C, D) = \top$ also.
- Lines 4 to 10: this case does not apply since it only applies when $\text{size}(C) \geq 3$.
- Lines 11 and 12: this case does not apply since it only applies when $\text{size}(D) \geq 3$.
- Lines 13 to 19: this case does not apply since it only applies when $\text{size}(C) + \text{size}(D) \geq 4$.
- Lines 20 and 21:
In this case, $C \neq \top$, $C \neq D$, C and D are not conjunctions and C and D are not existential restrictions with the same initial role. Thus C and D can be either existential restrictions with a different initial role and/or concept names. In any case, this leads to $\text{br}_C \cap \text{br}_D = \emptyset$, $\text{subd}_C \cap \text{subd}_D = \emptyset$ and there is no couple of branches that begin with the same role. Thus $\text{br} = \text{br}_C$ with

$$\text{br} = \text{br}_C \setminus (\text{br}_D \cup \{S = \exists r_1. \exists r_2 \dots \exists r_n. \top \in \text{br}_C, n \geq 0 \mid$$

$$\exists S' = \exists r_1 \dots \exists r_n. \exists r_{n+1} \dots \exists r_{n+m}. P \in \text{br}_D, m \geq 0\})$$
So $C \Delta D = C$. Besides, $\text{tso}(C, D) = C$ also.

General case: $s_{in} = \text{size}(C) + \text{size}(D) = n_r + 1$, with $n_r \geq 3$

We recall in this case, the induction hypothesis (IH) says: $\forall C'$ and D' with $s'_{in} = \text{size}(C') + \text{size}(D') \leq n_r$, $\text{tso}(C', D') = C' \Delta D'$.

- Lines 1 and 2:
 - If $C = D = \top$, this case does not apply since $\text{size}(C) + \text{size}(D) = 2$.
 - If $C = \top$ (and $C \neq D$), $\text{br} = \{\top\} \setminus (\text{br}_D \cup \{\top\}) = \emptyset$ and thus $C \Delta D = \top$, and $\text{tso}(C, D) = \top$ also.
 - If $C = D$ (and $C \neq \top$), $\text{br} = \text{br}_C \setminus (\text{br}_C \cup \emptyset) = \emptyset$ and thus $C \Delta D = \top$, and $\text{tso}(C, D) = \top$ also.
- Lines 4 to 10: in this case, we have $C = \bigwedge_{i=1}^k C_i$, with $k \geq 2$ and each C_i is not a conjunction. So $\text{br}_C = \bigcup_{i=1}^k \text{br}_{C_i}$. Moreover, since $\text{size}(C_i) \leq \text{size}(C) - 2$ we have $\text{size}(C_i) + \text{size}(D) \leq n_r - 2$. Thus, by IH, there is:
$$\textcircled{b} \forall i \in \{1, \dots, k\},$$

$$\text{tso}(C_i, D) = C_i \Delta D$$

$$= \begin{cases} E_i \mid \textcircled{1} E_i \in \text{subd}_{C_i} \text{ and } \textcircled{2} \text{br}_{E_i} = \text{br}_i & \text{if } \text{br}_i \neq \emptyset \\ \top & \text{if } \text{br}_i = \emptyset \end{cases}$$

with $\text{br}_i = \text{br}_{C_i} \setminus (\text{br}_D \cup \{S = \exists r_1. \exists r_2 \dots \exists r_n. \top \in \text{br}_{C_i}, n \geq 0 \mid$

$$\exists S' = \exists r_1 \dots \exists r_n. \exists r_{n+1} \dots \exists r_{n+m}. P \in \text{br}_D, m \geq 0\})$$

Let's study $E = \bigwedge_{i=1}^k \text{tso}(C_i, D)$ since the output of the algorithm is built from it (modulo lines 7 and 9). Thus $\text{br}_E = \text{br}_{\bigwedge_{i=1}^k \text{tso}(C_i, D)}$. According to \textcircled{b} , there is:

$\forall 1 \leq i \leq k,$

$$\text{br}_{\text{tso}(C_i, D)} = \begin{cases} \text{br}_i & \text{if } \text{br}_i \neq \emptyset \\ \{\top\} & \text{if } \text{br}_i = \emptyset \end{cases}$$

with

$$\text{br}_i = \text{br}_{C_i} \setminus (\text{br}_D \cup \{S = \exists r_1 \dots \exists r_n. \top \in \text{br}_{C_i} \mid \exists S' = \exists r_1 \dots \exists r_{n+m}. P \in \text{br}_D\})$$

So:

$$\begin{aligned} \text{br}_E &= \text{br}_{\prod_{i=1}^k \text{tso}(C_i, D)} = \bigcup_{i=1}^k \text{br}_{\text{tso}(C_i, D)} \\ &= \begin{cases} \bigcup_{i=1}^k \text{br}_i & \text{if } \bigcup_{i=1}^k \text{br}_i \neq \emptyset \\ \{\top\} & \text{if } \bigcup_{i=1}^k \text{br}_i = \emptyset \end{cases} \end{aligned}$$

Now, let's note $\bigcup_{i=1}^k \text{br}_i = \text{br}$. We have:

$$\begin{aligned} \text{br} &= \bigcup_{i=1}^k \left(\text{br}_{C_i} \setminus (\text{br}_D \cup \{S = \exists r_1 \dots \exists r_n. \top \in \text{br}_{C_i} \mid \exists S' = \exists r_1 \dots \exists r_{n+m}. P \in \text{br}_D\}) \right) \\ &= \left(\bigcup_{i=1}^k \text{br}_{C_i} \right) \setminus (\text{br}_D \cup \bigcup_{i=1}^k \{S = \exists r_1 \dots \exists r_n. \top \in \text{br}_{C_i} \mid \exists S' = \exists r_1 \dots \exists r_{n+m}. P \in \text{br}_D\}) \\ &= \left(\bigcup_{i=1}^k \text{br}_{C_i} \right) \setminus (\text{br}_D \cup \{S = \exists r_1 \dots \exists r_n. \top \in \left(\bigcup_{i=1}^k \text{br}_{C_i} \right) \mid \exists S' = \exists r_1 \dots \exists r_{n+m}. P \in \text{br}_D\}) \end{aligned}$$

And thanks to ⑥, this leads to:

$$\text{br} = \text{br}_C \setminus (\text{br}_D \cup \{S = \exists r_1 \dots \exists r_n. \top \in \text{br}_C \mid \exists S' = \exists r_1 \dots \exists r_{n+m}. P \in \text{br}_D\})$$

and

$$\text{br}_E = \begin{cases} \text{br} & \text{if } \text{br} \neq \emptyset \\ \{\top\} & \text{if } \text{br} = \emptyset \end{cases}$$

This is the proof of ② for E .

Now, since $E = \prod_{i=1}^k \text{tso}(C_i, D)$, $C = \prod_{i=1}^k C_i$ and, according to ⑥, $\text{tso}(C_i, D) \in \text{subd}_{C_i}$, we derive $E \in \text{subd}_C$. This is the proof of ① for E (when $\text{br} \neq \emptyset$).

- Lines 11 and 12: in this case, we have $D = \prod_{j=1}^m D_j$, with $m \geq 2$ and each D_j is not a conjunction. We have then $\text{size}(D) = (\sum_{j=1}^m \text{size}(D_j) + (m - 1))$. Besides, the output of $\text{tso}(C, D)$ is set to be

$$E = \text{tso}(\text{tso}(\dots \text{tso}(\text{tso}(C, D_1), D_2), \dots), D_{m-1}), D_m).$$

We recall we have shown algorithm 1 always terminates with $\text{size}(\text{tso}(C, D)) < \text{size}(C) + \text{size}(D)$. Applied to E this means there is:

$$\text{size}(\text{tso}(C, D_1)) \leq \text{size}(C) + \text{size}(D_1) - 1$$

$$\text{size}(\text{tso}(\text{tso}(C, D_1), D_2)) \leq \text{size}(\text{tso}(C, D_1)) + \text{size}(D_2) - 1$$

...

$$\text{size}(\text{tso}(\text{tso}(\dots, D_{m-2}), D_{m-1})) \leq \text{size}(\text{tso}(\dots, D_{m-2})) + \text{size}(D_{m-1}) - 1$$

Thus, $\forall k \in \{2, \dots, m - 1\}$:

$$\text{size}(\text{tso}(\text{tso}(\dots, D_{k-1}), D_k)) \leq \text{size}(C) + (\sum_{j=1}^k \text{size}(D_j)) - k$$

Since $\text{size}(D) = (\sum_{j=1}^m \text{size}(D_j) + (m - 1))$, $m \geq 2$ and $s_{in} = \text{size}(C) + \text{size}(D) = n_r + 1$,

we have $\forall k \in \{2, \dots, m - 1\}$:

$$\text{size}(\text{tso}(\text{tso}(\dots, D_{k-1}), D_k)) + \text{size}(D_{k+1})$$

$$\begin{aligned} &\leq \text{size}(C) + (\sum_{j=1}^{k+1} \text{size}(D_j)) - k \\ &< \text{size}(C) + \text{size}(D) = s_{in} = n_r + 1 \end{aligned}$$

So we can apply the IH to get:

$$\text{tso}(C, D_1) = C \Delta D_1$$

$$\text{tso}(\text{tso}(C, D_1), D_2) = \text{tso}(C, D_1) \Delta D_2$$

...

$$\text{tso}(\text{tso}(\dots, D_{m-2}), D_{m-1}) = \text{tso}(\dots, D_{m-2}) \Delta D_{m-1}$$

$$E = \text{tso}(\text{tso}(\dots, D_{m-1}), D_m) = \text{tso}(\dots, D_{m-1}) \Delta D_m$$

For a sake of clarity, we rename $\text{tso}(\text{tso}(\dots, D_{k-1}), D_k)$ as tso_k , for all $k \in \{2, \dots, m\}$.

This leads to :

$$\text{tso}_1 = C \Delta D_1$$

$$\text{tso}_2 = \text{tso}_1 \Delta D_2$$

...

$$\text{tso}_{m-1} = \text{tso}_{m-2} \Delta D_{m-1}$$

$$E = \text{tso}_m = \text{tso}_{m-1} \Delta D_m$$

Using the characterization of $C \Delta D$ obtained at step 2, this leads to:

$$\begin{aligned} \text{tso}_1 &= \begin{cases} E_1 \mid \textcircled{1} E_1 \in \text{subd}_C \text{ and } \textcircled{2} \text{br}_{E_1} = \text{br}_1 & \text{if } \text{br}_1 \neq \emptyset \\ \top & \text{if } \text{br}_1 = \emptyset \end{cases} \\ &\quad \text{with } \text{br}_1 = \text{br}_C \setminus (\text{br}_{D_1} \cup \{S = \exists r_1. \exists r_2 \dots \exists r_n. \top \in \text{br}_C, n \geq 0 \mid \\ &\quad \exists S' = \exists r_1 \dots \exists r_n. \exists r_{n+1} \dots \exists r_{n+m}. P \in \text{br}_{D_1}, m \geq 0\}) \\ \text{tso}_2 &= \begin{cases} E_2 \mid \textcircled{1} E_2 \in \text{subd}_{\text{tso}_1} \text{ and } \textcircled{2} \text{br}_{E_2} = \text{br}_2 & \text{if } \text{br}_2 \neq \emptyset \\ \top & \text{if } \text{br}_2 = \emptyset \end{cases} \\ &\quad \text{with } \text{br}_2 = \text{br}_{\text{tso}_1} \setminus (\text{br}_{D_2} \cup \{S = \exists r_1. \exists r_2 \dots \exists r_n. \top \in \text{br}_{\text{tso}_1}, n \geq 0 \mid \\ &\quad \exists S' = \exists r_1 \dots \exists r_n. \exists r_{n+1} \dots \exists r_{n+m}. P \in \text{br}_{D_2}, m \geq 0\}) \\ &\quad \dots \\ E &= \text{tso}_m \\ &= \begin{cases} E_m \mid \textcircled{1} E_m \in \text{subd}_{\text{tso}_{m-1}} \text{ and } \textcircled{2} \text{br}_{E_m} = \text{br}_m & \text{if } \text{br}_m \neq \emptyset \\ \top & \text{if } \text{br}_m = \emptyset \end{cases} \\ &\quad \text{with } \text{br}_m = \text{br}_{\text{tso}_{m-1}} \setminus (\text{br}_{D_m} \cup \{S = \exists r_1. \exists r_2 \dots \exists r_n. \top \in \text{br}_{\text{tso}_{m-1}}, n \geq 0 \mid \\ &\quad \exists S' = \exists r_1 \dots \exists r_n. \exists r_{n+1} \dots \exists r_{n+m}. P \in \text{br}_{D_m}, m \geq 0\}) \end{aligned}$$

Since $\text{br}_D = \bigcup_{j=1}^m \text{br}_{D_j}$, it follows straightforwardly that, if $\text{br} \neq \emptyset$, then $\textcircled{1} E \in \text{subd}_C$ and $\textcircled{2} \text{br}_E = \text{br}$, and if $\text{br} = \emptyset$, then $E = \top$, with $\text{br} = \text{br}_C \setminus (\text{br}_D \cup \{S = \exists r_1. \exists r_2 \dots \exists r_n. \top \in \text{br}_C, n \geq 0 \mid \exists S' = \exists r_1 \dots \exists r_n. \exists r_{n+1} \dots \exists r_{n+m}. P \in \text{br}_D, m \geq 0\})$, i.e. $\text{tso}(C, D) = E = C \Delta D$.

- Lines 13 to 19: in this case, $C = \exists r. C'$ and $D = \exists r. D'$.

Clearly $\text{size}(C') + \text{size}(D') = \text{size}(C) + \text{size}(D) - 2 \leq n_r$. So the IH can be applied:

$$\begin{aligned} \text{tso}(C', D') &= C' \Delta D' = \begin{cases} E' \mid \textcircled{1} E' \in \text{subd}_{C'} \text{ and } \textcircled{2} \text{br}_{E'} = \text{br}' & \text{if } \text{br}' \neq \emptyset \\ \top & \text{if } \text{br}' = \emptyset \end{cases} \\ &\quad \text{with } \text{br}' = \text{br}_{C'} \setminus (\text{br}_{D'} \cup \{S = \exists r_1. \exists r_2 \dots \exists r_n. \top \in \text{br}_{C'}, n \geq 0 \mid \\ &\quad \exists S' = \exists r_1 \dots \exists r_n. \exists r_{n+1} \dots \exists r_{n+m}. P \in \text{br}_{D'}, m \geq 0\}) \end{aligned}$$

Besides, it is also clear that $\text{br}_C = \{\exists r. S \mid S \in \text{br}_{C'}\}$ and $\text{br}_D = \{\exists r. S \mid S \in \text{br}_{D'}\}$. Thus if we define:

$$\begin{aligned} \text{br} &= \text{br}_C \setminus (\text{br}_D \cup \{S = \exists r_1. \exists r_2 \dots \exists r_n. \top \in \text{br}_C, n \geq 0 \mid \\ &\quad \exists S' = \exists r_1 \dots \exists r_n. \exists r_{n+1} \dots \exists r_{n+m}. P \in \text{br}_D, m \geq 0\}) \end{aligned}$$

then $\text{br}' = \emptyset$ iff $\text{br} = \emptyset$.

Moreover, we also have $\text{subd}_C = \{\exists r.S \mid S \in \text{subd}_{C'}\}$ and $\text{subd}_D = \{\exists r.S \mid S \in \text{subd}_{D'}\}$. Thus, when $\text{br}' \neq \emptyset$, we conclude the concept E' that follows ①' $E' \in \text{subd}_{C'}$ and ② $\text{br}_{E'} = \text{br}'$ defines a unique concept $E = \exists r.E$ that follows ①' $E \in \text{subd}_C$ and ② $\text{br}_E = \text{br}$. This shows $\text{tso}(C, D) = C \Delta D$ for lines 16 and 18.

- Lines 20 and 21:

In this case, $C \neq \top$, $C \neq D$, C and D are not conjunctions and C and D are not existential restrictions with the same initial role. Thus C and D can be either existential restrictions with a different initial role and/or (different) concept names. In any case, this leads to $\text{br}_C \cap \text{br}_D = \emptyset$, $\text{subd}_C \cap \text{subd}_D = \emptyset$ and there is no couple of branches that begin with the same role. Thus $\text{br} = \text{br}_C$ with

$$\text{br} = \text{br}_C \setminus (\text{br}_D \cup \{S = \exists r_1.\exists r_2...\exists r_n.\top \in \text{br}_C, n \geq 0 \mid \exists S' = \exists r_1...\exists r_n.\exists r_{n+1}...\exists r_{n+m}.P \in \text{br}_D, m \geq 0\})$$

So $C \Delta D = C$. Besides, $\text{tso}(C, D) = C$ also.

d. PTIME computational complexity In this proof, we assume that k is the constant execution time of elementary operations: concatenations of two concepts A and B with a \sqcap symbol to obtain $A \sqcap B$, concatenations of an existential quantification $\exists r.$ with a concept A to obtain $\exists r.A$, variable assignments, comparisons of two concept or role names, and recursive calls. We also assume that checking whether 2 concepts of sizes n_1 and n_2 are syntactically identical can be done in linear time $\mathcal{O}(\text{Min}(n_1, n_2))$. Besides, checking whether a concept is a conjunction or an existential restriction can be done in constant time since we suppose concepts are represented/stored as syntactical trees.

Let C and D be two \mathcal{EL} concepts. For a sake of clarity, we note $\text{size}(C) = n_C$ and $\text{size}(D) = n_D$. We also note $n = n_C + n_D$. When C is a conjunction, we consider it has p conjuncts (and not n as in algorithm 1 since n is already equal to $n_C + n_D$). When D is a conjunction, we consider it has m conjuncts (as in algorithm 1). Let's find an upper bound for $T(n)$ which is the execution time of algorithm tso on an input of size n , i.e. of the call $\text{tso}(C, D)$.

We first recall algorithm 1 which computes $\text{tso}(C, D)$.

Require: C and D two \mathcal{EL} concepts.

Ensure: $C \Delta D$ (cf. def. 8)

```

1: if  $C = D$  or  $C = \top$  then
2:    $\text{Result} := \top$                                      {Case 1 and case 2}
3: else
4:   if  $C = C_1 \sqcap \dots \sqcap C_n$  with  $n \geq 2$  then
5:      $\text{Result1} := \text{tso}(C_1, D) \sqcap \dots \sqcap \text{tso}(C_n, D)$ 
6:     if There is at least one conjunct  $\neq \top$  in  $\text{Result1}$  then
7:        $\text{Result} := \text{Result1}$  without any  $\top$  conjunct.           {Case 3}
8:     else
9:        $\text{Result} := \top$                                            {Case 4}
10:    end if
11:  else if  $D = D_1 \sqcap \dots \sqcap D_m$  with  $m \geq 2$  then
12:     $\text{Result} := \text{tso}(\dots (\text{tso}(\text{tso}(C, D_1), D_2), \dots), D_m)$            {Case 5}
```

```

13: else if  $C = \exists r.C'$  and  $D = \exists r.D'$  then
14:    $Result1 := \text{tso}(C', D')$ 
15:   if  $Result1 = \top$  then
16:      $Result := \top$  {Case 6}
17:   else
18:      $Result := \exists r.Result1$  {Case 7}
19:   end if
20: else
21:    $Result := C$  {Case 8}
22: end if
23: end if
24: return  $Result$ 

```

Since algorithm 1 is made of nested tests leading to 8 possible cases (as shown above), we have $T(n)$ is less or equal to the maximum of the following 8 cases:

$Min(n_C, n_D) * k$ $+k,$	Line 1: test $C = D$ Line 2: variable assignment	} case 1
$(Min(n_C, n_D) + 1) * k$ $+k,$	L1: test $C = D$ and $C = \top$ L2: variable assignment	} case 2
$(Min(n_C, n_D) + 1) * k$ $+k$ $+p * k$ $+\sum_{i=1}^p T(\text{size}(C_i) + n_D)$ $+(p-1) * k$ $+k$ $+p * k$ $+(p-1) * k$ $+k,$	L1: $C = D$ and $C = \top$ L4: test $C = C_1 \sqcap \dots \sqcap C_p$ L5: p recursive calls L5: recursive calls execution time L5: building the answer by concatenating with \sqcap L5: variable assignment L6: p tests conjuncts $\neq \top$ L7: removal of at worst $p-1$ conjuncts L7: variable assignment	} case 3
$(Min(n_C, n_D) + 1) * k$ $+k$ $+p * k$ $+\sum_{i=1}^p T(\text{size}(C_i) + n_D)$ $+(p-1) * k$ $+k$ $+p * k$ $+k,$	L1: $C = D$ and $C = \top$ L4: test $C = C_1 \sqcap \dots \sqcap C_p$ L5: p recursive calls L5: recursive calls execution time L5: building the answer by concatenating with \sqcap L5: variable assignment L6: p tests conjuncts $\neq \top$ L9: variable assignment	} case 4

$(Min(n_C, n_D) + 1) * k$ $+k$ $+k$ $+m * k$ $+T(n_C + size(D_1))$ $+T(size(tso(C, D_1)) + size(D_2))$ $+...$ $+T(size(tso(...)) + size(D_{m-1}))$ $+T(size(tso(...)) + size(D_m))$ $+k,$	L1: $C = D$ and $C = \top$ L4: test $C = C_1 \sqcap ... \sqcap C_p$ L11: $D = D_1 \sqcap ... \sqcap D_m$ L12: m recursive calls L12: recursive calls execution time L12: affectation	} case 5
$(Min(n_C, n_D) + 1) * k$ $+k$ $+k$ $+2 * k$ $+k$ $+T(n_C - 1 + n_D - 1)$ $+k$ $+k$ $+k,$	L1: $C = D$ and $C = \top$ L4: test $C = C_1 \sqcap ... \sqcap C_p$ L11: $D = D_1 \sqcap ... \sqcap D_m$ L13: $C = \exists r.C'$ and $D = \exists r.D'$ L14: recursive call L14: recursive call execution time L14: variable assignment L15: test $Result = \top$ L16: variable assignment	} case 6
$(Min(n_C, n_D) + 1) * k$ $+k$ $+k$ $+2 * k$ $+k$ $+T(n_C - 1 + n_D - 1)$ $+k$ $+k$ $+k$ $+k$	L1: $C = D$ and $C = \top$ L4: test $C = C_1 \sqcap ... \sqcap C_p$ L11: $D = D_1 \sqcap ... \sqcap D_m$ L13: $C = \exists r.C'$ and $D = \exists r.D'$ L14: recursive call L14: recursive call execution time L14: variable assignment L15: test $Result = \top$ L18: building the answer by concatenating with $\exists r$. L18. variable assignment	} case 7
$(Min(n_C, n_D) + 1) * k$ $+k$ $+k$ $+2 * k$ $+k$	L1: $C = D$ and $C = \top$ L4: test $C = C_1 \sqcap ... \sqcap C_p$ L11: $D = D_1 \sqcap ... \sqcap D_m$ L13: $C = \exists r.C'$ and $D = \exists r.D'$ L21. variable assignment	} case 8

After simplification, we get:

$$T(n) \leq Max(\begin{array}{ll} (Min(n_C, n_D) + 1) * k, & \} \text{ case 1} \\ (Min(n_C, n_D) + 2) * k, & \} \text{ case 2} \\ (Min(n_C, n_D) + 2 + 4p) * k + \sum_{i=1}^p T(size(C_i) + n_D), & \} \text{ case 3} \end{array}$$

$$\begin{array}{l}
(Min(n_C, n_D) + 3 + 3p) * k + \sum_{i=1}^p T(\text{size}(C_i) + n_D), \quad \left. \begin{array}{l} \\ \\ \\ \\ \\ \end{array} \right\} \text{case 4} \\
(Min(n_C, n_D) + 4 + m) * k \\
+ T(n_C + \text{size}(D_1)) \\
+ T(\text{size}(\text{tso}(C, D_1)) + \text{size}(D_2)) \\
+ \dots \\
+ T(\text{size}(\text{tso}(\dots)) + \text{size}(D_{m-1})) \\
+ T(\text{size}(\text{tso}(\dots)) + \text{size}(D_m)) \quad \left. \begin{array}{l} \\ \\ \\ \end{array} \right\} \text{case 5} \\
(Min(n_C, n_D) + 9) * k + T(n - 2) \quad \left. \begin{array}{l} \\ \end{array} \right\} \text{case 6} \\
(Min(n_C, n_D) + 10) * k + T(n - 2) \quad \left. \begin{array}{l} \\ \end{array} \right\} \text{case 7} \\
(Min(n_C, n_D) + 6) * k \quad \left. \begin{array}{l} \\ \end{array} \right\} \text{case 8} \\
)
\end{array}$$

We can see that there are cases with some recursive calls and cases without recursive calls. Cases without recursive calls execute in linear time $\mathcal{O}(n)$ since $n_C \leq n$ and $n_D \leq n$. Cases with recursive calls have a linear time part and a recursive call part. Thus, a worst case happens when the number of recursive calls is maximal. Within cases having recursive calls, there are:

- cases 3 and 4 having p recursive calls (p is the number of conjuncts of C),
- case 5 having m recursive calls (m is the number of conjuncts of D)
- and cases 6 and 7 having a single recursive call.

As p and m are both greater than 2, then maximizing the number of recursive calls implies getting into cases 3, 4 or 5. In other terms, for a given input size n , we have to maximize p and m , which in turn implies to minimize existential restrictions. This means that a worst case for algorithm 1 happens when C and D are both conjunctions of concepts names, without any existential restrictions. Moreover, in this worst case, concepts names in C and D must all be pairwise distinct and different from \top in order not to go through case 1 and stop. It is easy to see that we then have $\text{tso}(C, D) = C$.

In this worst case, the execution runs as follows:

- the $\text{tso}(C, D)$ main call is handled by case 3, i.e. goes through lines 4, 5, 6 and 7.
- line 5 triggers p recursive calls $\text{tso}(C_i, D)$, $1 \leq i \leq p$.
 - each recursive call $\text{tso}(C_i, D)$ is handled by case 5, i.e. goes through lines 1, 4, 11 and 12,
 - line 12 triggers m recursive calls $\text{tso}(C_i, D_j)$, $1 \leq j \leq m$. For all m recursive calls, the first argument stays C_i since C is not modified in the whole process.
 - * each recursive call $\text{tso}(C_i, D_j)$ is handled by case 8, i.e. goes through lines 1, 4, 11, 13 and 21.

As C is a conjunction of p concept names (of size 1), there is: $n_C = \text{size}(C) = (\sum_{i=1}^p 1) + p - 1 = 2p - 1$. In the same way: $n_D = \text{size}(D) = (\sum_{j=1}^m 1) + m - 1 = 2m - 1$. As $n = n_C + n_D$, we have $n = 2p + 2m - 2$ and thus $p + m = n/2 + 1$. We can now evaluate the execution time of $\text{tso}(C, D)$ in the worst case described above:

$$T(n) = (Min(n_C, n_D) + 2 + 4p) * k + \sum_{i=1}^p T(\text{size}(C_i) + n_D)$$

with

- for all i in $\{1, \dots, p\}$:
 $T(\text{size}(C_i) + n_D) = (\text{Min}(\text{size}(C_i), \text{size}(D)) + 4 + m) * k$
 $+T(\text{size}(C_i) + \text{size}(D_1))$
 $+T(\text{size}(C_i) + \text{size}(D_2))$
 $+\dots$
 $+T(\text{size}(C_i) + \text{size}(D_{m-1}))$
 $+T(\text{size}(C_i) + \text{size}(D_m))$
- for all j in $\{1, \dots, m\}$:
 $T(\text{size}(C_i) + \text{size}(D_j)) = (\text{Min}(\text{size}(C_i), \text{size}(D_j)) + 6) * k$

By replacing $\text{size}(C_i)$ and $\text{size}(D_j)$ by 1 since they are concepts names, we get:

$$\begin{aligned} T(n) &= (\text{Min}(n_C, n_D) + 2 + 4p) * k + p * (5 + 8m) * k \\ &= k * \text{Min}(2p - 1, 2m - 1) + 4k * p + 2k + 5k * p + 8k * mp \end{aligned}$$

Since k is constant, $T(n)$ is in $\mathcal{O}(mp)$. As $m + p = n/2 + 1$, mp is maximal when $m = p = n/4 + 1/2$. This enables us to deduce $T(n)$ is in $\mathcal{O}(n^2)$. □

.3. Proof of Proposition 2

Proposition 2. Let \mathcal{T} be a TBox and $(C, D) \in (\mathcal{T}_{\mathcal{EL}})^2$. There is:

- $C \ominus_{\mathcal{T}} D$ exists and is unique (up to $\equiv_{\mathcal{T}}$).
- $C \ominus_{\mathcal{T}} D = \mathcal{T}^*(C) \Delta \mathcal{T}^*(D) = \text{cso}(\mathcal{T}, C, D)$ (soundness).
- Computing $\text{cso}(\mathcal{T}, C, D)$ is in EXPTIME in the sizes of \mathcal{T} , C and D and in PTIME in the sizes of $\mathcal{T}^*(C)$ and $\mathcal{T}^*(D)$.

Proof.

a. Existence and unicity In the case where $\text{char}_{\mathcal{T}}^{\mathcal{T}} C \subseteq \text{dcom}_{\mathcal{T}}^{\mathcal{T}} C, D$, existence and unicity of $C \ominus_{\mathcal{T}} D$ are trivial. Otherwise, $\text{char}_{\mathcal{T}}^{\mathcal{T}} C \not\subseteq \text{dcom}_{\mathcal{T}}^{\mathcal{T}} C, D$ is equivalent to $\exists S \in \text{char}_{\mathcal{T}}^{\mathcal{T}} C \mid D \not\sqsubseteq_{\mathcal{T}} S$. So there always exists $E = S$ such that $C \sqsubseteq_{\mathcal{T}} E$ and $\text{dcom}_{\mathcal{T}}^{\mathcal{T}} E, D = \emptyset$. Thus $C \ominus_{\mathcal{T}} D$ exists. Now, since $C \ominus_{\mathcal{T}} D$ must be minimal w.r.t. to $\sqsubseteq_{\mathcal{T}}$, it is easy to see it is unique since it suffices to take the conjunction of all elements of $\{E \in \mathcal{T}_{\mathcal{EL}} \mid C \sqsubseteq_{\mathcal{T}} E \text{ and } \text{dcom}_{\mathcal{T}}^{\mathcal{T}} E, D = \emptyset\}$ to have the minimal one. Since defined concepts can be expanded w.r.t. \mathcal{T} , unicity is up to $\equiv_{\mathcal{T}}$.

b. Soundness Here, we have to show $C \ominus_{\mathcal{T}} D = \mathcal{T}^*(C) \Delta \mathcal{T}^*(D)$ with:

$$C \ominus_{\mathcal{T}} D = \begin{cases} \text{Min}_{\sqsubseteq_{\mathcal{T}}} \{E \in \mathcal{T}_{\mathcal{EL}} \mid C \sqsubseteq_{\mathcal{T}} E \text{ and } \text{dcom}_{\mathcal{T}}^{\mathcal{T}} E, D = \emptyset\} & \text{if } \text{char}_{\mathcal{T}}^{\mathcal{T}} C \not\subseteq \text{dcom}_{\mathcal{T}}^{\mathcal{T}} C, D \\ \top & \text{if } \text{char}_{\mathcal{T}}^{\mathcal{T}} C \subseteq \text{dcom}_{\mathcal{T}}^{\mathcal{T}} C, D \end{cases}$$

and

$$\mathcal{T}^*(C) \Delta \mathcal{T}^*(D) = \begin{cases} \text{Min}_{\sqsubseteq} \{E \in \mathcal{T}_{\mathcal{EL}}^{\text{prim}} \mid \mathcal{T}^*(C) \sqsubseteq E \text{ and } \text{br}_E = \text{br}\} & \text{if } \text{br} \neq \emptyset \\ \top & \text{if } \text{br} = \emptyset \end{cases}$$

with $\text{br} = \text{br}_{\mathcal{T}^*(C)} \setminus (\text{br}_{\mathcal{T}^*(D)} \cup \{S = \exists r_1. \exists r_2 \dots \exists r_n. \top \in \text{br}_{\mathcal{T}^*(C)}, n \geq 0 \mid \exists S' = \exists r_1 \dots \exists r_n. \exists r_{n+1} \dots \exists r_{n+m}. P \in \text{br}_{\mathcal{T}^*(D)}, m \geq 0\})$

and with P any concept name or \top (except if $m=0$).

Remark: $E \in \mathcal{T}_{\mathcal{EL}}^{\text{prim}}$ as a consequence of $\mathcal{T}^*(C) \sqsubseteq E$.

First we show: $(\star) \text{ char}_C^{\mathcal{T}} \subseteq \text{dcom}_{C,D}^{\mathcal{T}} \Leftrightarrow \text{br} = \emptyset \ (\star\star)$.

\Rightarrow :

Let $S \in \text{br}_{\mathcal{T}^*(C)}$.

- Case 1: $S \in \text{char}_C^{\mathcal{T}}$.

Thus $S \in \text{br}_{\mathcal{T}}^{\text{prim}}$. Besides, since $(\star) \Leftrightarrow \forall S \in \text{char}_C^{\mathcal{T}}, D \sqsubseteq_{\mathcal{T}} S$, there is $D \sqsubseteq_{\mathcal{T}} S$. Since $S \in \text{br}_{\mathcal{T}}^{\text{prim}}$, we easily have $\mathcal{T}^*(D) \sqsubseteq S$. By corollary 1, this is equivalent to the fact that there exists $S' \in \text{subd}_{\mathcal{T}^*(D)}$ such that S' can be obtained from S by applying anywhere in S zero or many times the following syntactic rules (r1) and (r2) that amount to replacing their left hand side by their right hand side:

(r1) $\exists r.G \sqcap \exists r.H \rightsquigarrow \exists r.(G \sqcap H)$

(r2) $\top \rightsquigarrow H$

with r any role, and G and H any concepts. For a sake of simplicity, from now on, we note these transformations : $S \xrightarrow{(r1),(r2)} S'$. As S is a branch, S' must also be a branch. So there is: $\exists S' \in \text{br}_{\mathcal{T}^*(D)} \mid S \xrightarrow{(r1),(r2)} S'$. Now, since S is a branch, (r1) cannot be applied. So $\exists S' \in \text{br}_{\mathcal{T}^*(D)} \mid S \xrightarrow{(r2)} S'$. Since (r1) can only be applied once to a branch, we have the two following cases:

- either $S = S'$
- or $S = \exists r_1 \dots \exists r_n. \top$ and $S' = \exists r_1 \dots \exists r_n. H$, with $n \geq 0$ and $H \in \text{br}_{\mathcal{T}}^{\text{prim}}$.

By the definition of br , $S \notin \text{br}$ in both cases.

- Case 2: $S \notin \text{char}_C^{\mathcal{T}}$.

Thus $\forall G \in \mathcal{T}_{\mathcal{EL}} \mid G \equiv_{\mathcal{T}} C$ and $S \in \text{br}_G$, $\exists H \in \mathcal{T}_{\mathcal{EL}} \mid \text{br}_H = \text{br}_G \setminus \{S\}$ and $H \equiv_{\mathcal{T}} C$. As $S \in \text{br}_{\mathcal{T}^*(C)}$, it means that the concept $\mathcal{T}^*(C)^{-S}$, defined with $\text{br}_{\mathcal{T}^*(C)^{-S}} = \text{br}_{\mathcal{T}^*(C)} \setminus \{S\}$ and $\mathcal{T}^*(C)^{-S} \equiv \mathcal{T}^*(C)$, exists, is unique and is such that $\mathcal{T}^*(C)^{-S} \sqsubseteq S$. Following the same kind of reasoning as in previous case, we then derive:

$\exists S' \in \text{br}_{\mathcal{T}^*(C)^{-S}} \mid$

- either $S = S'$
- or $S = \exists r_1 \dots \exists r_n. \top$ and $S' = \exists r_1 \dots \exists r_n. H$, with $n \geq 0$ and $H \in \text{br}_{\mathcal{T}}^{\text{prim}}$.

Since the complete expansion of a concept C w.r.t. \mathcal{T} is a unique concept $\mathcal{T}^*(C)$, we see that $\text{br}_{\mathcal{T}^*(C)} = \text{char}_{\mathcal{T}^*(C)}^{\emptyset} \subseteq \text{char}_C^{\mathcal{T}}$. This means that $S' \in \text{char}_{\mathcal{T}^*(C)^{-S}}^{\emptyset}$. So the case where $S = S'$ is not possible otherwise we would have $S \in \text{char}_C^{\mathcal{T}}$. Thus $S = \exists r_1 \dots \exists r_n. \top$ and $S' = \exists r_1 \dots \exists r_n. H$, with $n \geq 0$ and $H \in \text{br}_{\mathcal{T}}^{\text{prim}}$. Since $S' \in \text{char}_{\mathcal{T}^*(C)^{-S}}^{\emptyset}$ we have $S' \in \text{char}_C^{\mathcal{T}}$. Now we have supposed $(\star) \text{ char}_C^{\mathcal{T}} \subseteq \text{dcom}_{C,D}^{\mathcal{T}}$. So $S' \in \text{dcom}_{C,D}^{\mathcal{T}}$. So $D \sqsubseteq_{\mathcal{T}} S'$. Since $S' \in \text{br}_{\mathcal{T}}^{\text{prim}}$, we have $\mathcal{T}^*(D) \sqsubseteq S'$. A same reasoning as before, based on corollary 1, leads to: $\exists S'' \in \text{br}_{\mathcal{T}^*(D)} \mid S' = S''$. And then we derive $S \notin \text{br}$.

So, in both cases, $S \in \text{br}_{\mathcal{T}^*(C)}$ implies $S \notin \text{br}$. So $\text{br} = \emptyset$.

\Leftarrow :

We now suppose $\text{br} = \emptyset$. This means $\forall S \in \text{br}_{\mathcal{T}^*(C)}$, either ① $S \in \text{br}_{\mathcal{T}^*(D)}$, or ② $S = \exists r_1 \dots \exists r_n. \top$, with $n \geq 0$, and $\exists S' = \exists r_1 \dots \exists r_n \dots \exists r_{n+m}. P \in \text{br}_{\mathcal{T}^*(D)}$, with $m \geq 0$.

Let $S \in \text{char}_C^{\mathcal{T}}$. We have to show $S \in \text{dcom}_{C,D}^{\mathcal{T}}$. I.e. we have to show $D \sqsubseteq_{\mathcal{T}} S$. If ① is true, we have $\mathcal{T}^*(D) \sqsubseteq S$ and thus $D \sqsubseteq_{\mathcal{T}} S$. If ② is true, we have $\mathcal{T}^*(D) \sqsubseteq S' \sqsubseteq S$ and thus $D \sqsubseteq_{\mathcal{T}} S$.

So, we have shown $\text{char}_C^{\mathcal{T}} \subseteq \text{dcom}_{C,D}^{\mathcal{T}} \Leftrightarrow \text{br} = \emptyset$.

Now we prove a lemma that is useful for the second part of the soundness proof.

Lemma 1. *Let C and D be two concepts. The set br defined as follows:*

$$\text{br} = \text{br}_C \setminus (\text{br}_D \cup \{S = \exists r_1. \exists r_2 \dots \exists r_n. \top \in \text{br}_C, n \geq 0 \mid \exists S' = \exists r_1 \dots \exists r_n. \exists r_{n+1} \dots \exists r_{n+m}. P \in \text{br}_D, m \geq 0\})$$

is such that:

$$\text{br} = \{S \in \text{br}_C \mid D \not\sqsubseteq S\}$$

Proof.

$S \in \text{br}$

$$\Leftrightarrow \begin{aligned} &\bullet S \in \text{br}_C \text{ and} \\ &\bullet S \notin \text{br}_D \text{ and} \\ &\bullet \text{if } S = \exists r_1 \dots \exists r_n. \top, \text{ with } n \geq 0 \\ &\quad \text{then } \forall S' \in \text{br}_D, S' \neq \exists r_1 \dots \exists r_{n+m}. P, \text{ for all } m \geq 0 \text{ and } P \text{ concept} \\ &\quad \text{name or } \top. \end{aligned}$$

$$\Leftrightarrow \begin{aligned} &\bullet S \in \text{br}_C \text{ and} \\ &\forall S' \in \text{br}_D, S' \text{ cannot be obtained by applying zero or one time rule (r2) to } S. \end{aligned}$$

S is a branch

$$\Leftrightarrow \begin{aligned} &\bullet S \in \text{br}_C \text{ and} \\ &\forall S' \in \text{subd}_D, S' \text{ cannot be obtained by applying zero or many times rules} \\ &\text{(r1) and (r2) to } S. \end{aligned}$$

corollary 1

$$\Leftrightarrow \begin{aligned} &\bullet S \in \text{br}_C \text{ and} \\ &D \sqsubseteq_{\mathcal{T}} S \\ &\Leftrightarrow S \in \{T \in \text{br}_C \mid D \not\sqsubseteq T\} \end{aligned}$$

□

Before lemma 1, we have shown $\text{char}_C^{\mathcal{T}} \subseteq \text{dcom}_{C,D}^{\mathcal{T}} \Leftrightarrow \text{br} = \emptyset$. Now, assuming the opposite, i.e. $\text{char}_C^{\mathcal{T}} \not\subseteq \text{dcom}_{C,D}^{\mathcal{T}} \Leftrightarrow \text{br} \neq \emptyset$, we have to show:

$$\text{Min}_{\sqsubseteq_{\mathcal{T}}} \{E \in \mathcal{T}_{\mathcal{EL}} \mid C \sqsubseteq_{\mathcal{T}} E \text{ and } \text{dcom}_{E,D}^{\mathcal{T}} = \emptyset\} = \text{Min}_{\sqsubseteq} \{F \in \mathcal{T}_{\mathcal{EL}}^{\text{prim}} \mid \mathcal{T}^*(C) \sqsubseteq F \text{ and } \text{br}_F = \text{br}\}$$

Let $F \in \mathcal{T}_{\mathcal{EL}}^{\text{prim}}$ such that:

- ① $\mathcal{T}^*(C) \sqsubseteq F$ and
- ② $\text{br}_F = \text{br}$ and
- ③ F is minimal w.r.t. \sqsubseteq such that ① and ②.

We have to show that:

- Ⓐ $C \sqsubseteq_{\mathcal{T}} F$ and
- Ⓑ $\text{dcom}_{F,D}^{\mathcal{T}} = \emptyset$ and
- Ⓒ F is minimal w.r.t. $\sqsubseteq_{\mathcal{T}}$ such that Ⓐ and Ⓑ.

First, since $C \equiv_{\mathcal{T}} \mathcal{T}^*(C)$, Ⓐ implies $C \sqsubseteq_{\mathcal{T}} F$. This is Ⓐ.

Second, assuming $S \in \text{char}_F^{\mathcal{T}}$, we have to show $D \not\sqsubseteq_{\mathcal{T}} S$. As $F \in \mathcal{T}_{\mathcal{EL}}^{\text{prim}}$, we have $\text{char}_F^{\mathcal{T}} \subseteq \text{br}_F$. So $S \in \text{br}_F = \text{br}$. According to lemma 1, $S \in \text{br}_{\mathcal{T}^*(C)}$ and $\mathcal{T}^*(D) \not\sqsubseteq S$. Thus $D \not\sqsubseteq_{\mathcal{T}} S$. This is Ⓑ.

Third, Ⓒ \Leftrightarrow Ⓒ comes from $F \in \mathcal{T}_{\mathcal{EL}}^{\text{prim}}$.

c. Complexity The result trivially comes from complexity of the complete expansion and of algorithm 1.

□