

Additional material for DL2023 submission 4452 : 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, and show its tractability. CSO fits well with existential restrictions and does not require any subsumption relation between A and B to be defined and computed, 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. Normalization and classification in \mathcal{EL} [1, 2, 3]

Normalization and classification rules for \mathcal{EL} are given in table 1.


A TBox \mathcal{T} is in Baader normal form \mathcal{T}_{Bnf} after the three following steps: (i) exhaustively apply rule NF0, (ii) exhaustively apply rules NF1, NF2 and NF3, and (iii) exhaustively apply rules NF4 and NF5. In \mathcal{T}_{Bnf} , all axioms belong to one of the following types, with $C_1, C_2, D \in \mathbf{CT}$: $C_1 \sqsubseteq D$, $C_1 \sqcap C_2 \sqsubseteq D$, $C_1 \sqsubseteq \exists r.C_2$, or $\exists r.C_1 \sqsubseteq D$. This normalization is shown to be linear in the TBox size [1].

From a normalized TBox \mathcal{T}_{Bnf} , the classification procedure computes all subsumption links between couples of concept names w.r.t. \mathcal{T}_{Bnf} . More precisely, the complete classification of \mathcal{T}_{Bnf} generates sets $\mathbf{R}_{\mathcal{T}_{Bnf}}(V), \forall V \in \mathbf{r}_{\mathcal{T}_{Bnf}}$ and $\mathbf{S}_{\mathcal{T}_{Bnf}}(A), \forall A \in \mathbf{CT}_{\mathcal{T}_{Bnf}} \cup \{\top\}$, such that: (i) $B \in \mathbf{S}_{\mathcal{T}_{Bnf}}(A)$ if and only if $A \sqsubseteq_{\mathcal{T}_{Bnf}} B$ and (ii) $(A, B) \in \mathbf{R}_{\mathcal{T}_{Bnf}}(V)$ implies that $A \sqsubseteq_{\mathcal{T}_{Bnf}} \exists r.B$. The classification operates on \mathcal{T}_{Bnf} by the exhaustive application of rules CR1 to CR4, with $\mathbf{S}_{\mathcal{T}_{Bnf}}(A)$ initialized to $\{A, \top\}$ for all $A \in \mathbf{CT}_{\mathcal{T}_{Bnf}}$, and $\mathbf{R}_{\mathcal{T}_{Bnf}}(V)$ initialized to \emptyset for all $V \in \mathbf{r}_{\mathcal{T}_{Bnf}}$. This classification algorithm runs in PTIME in the size of \mathcal{T}_{Bnf} [1].

 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)

Table 1

Normalization (NF0 to NF5) and classification rules (CR1 to CR4) for \mathcal{EL} [1]. Assumptions for NF0 to NF5: $X, Y \notin \mathbf{C}_{\mathcal{T}}$ and A is a new concept name. Assumptions for CR1 to CR4: $C_1, C_2, C, D, C', D' \in \mathbf{C}_{\mathcal{T}}$ and $V \in \mathbf{R}_{\mathcal{T}}$.

NF0	$C \equiv D \rightarrow \{C \sqsubseteq D \text{ and } D \sqsubseteq C\}$	CR1	If $C' \in \mathbf{S}_{\mathcal{T}}(C), C' \sqsubseteq D \in \mathcal{T}$, and $D \notin \mathbf{S}_{\mathcal{T}}(C)$, then $\mathbf{S}_{\mathcal{T}}(C) := \mathbf{S}_{\mathcal{T}}(C) \cup \{D\}$
NF1	$C \sqcap Y \sqsubseteq E \rightarrow \{Y \sqsubseteq A, C \sqcap A \sqsubseteq E\}$	CR2	If $C_1, C_2 \in \mathbf{S}_{\mathcal{T}}(C), C_1 \sqcap C_2 \sqsubseteq D \in \mathcal{T}$ and $D \notin \mathbf{S}_{\mathcal{T}}(C)$, then $\mathbf{S}_{\mathcal{T}}(C) := \mathbf{S}_{\mathcal{T}}(C) \cup \{D\}$
NF2	$\exists r.X \sqsubseteq D \rightarrow \{X \sqsubseteq A, \exists r.A \sqsubseteq D\}$	CR3	If $C' \in \mathbf{S}_{\mathcal{T}}(C), C' \sqsubseteq \exists r.D \in \mathcal{T}$ and $(C, D) \notin \mathbf{R}_{\mathcal{T}}(V)$, then $\mathbf{R}_{\mathcal{T}}(V) := \mathbf{R}_{\mathcal{T}}(V) \cup \{(C, D)\}$
NF3	$X \sqsubseteq Y \rightarrow \{X \sqsubseteq A, A \sqsubseteq Y\}$	CR4	If $(C, D) \in \mathbf{R}_{\mathcal{T}}(V), D' \in \mathbf{S}_{\mathcal{T}}(D), \exists r.D' \sqsubseteq E \in \mathcal{T}$ and $E \notin \mathbf{S}_{\mathcal{T}}(C)$ then $\mathbf{S}_{\mathcal{T}}(C) := \mathbf{S}_{\mathcal{T}}(C) \cup \{E\}$
NF4	$B \sqsubseteq \exists r.X \rightarrow \{B \sqsubseteq \exists r.A, A \sqsubseteq X\}$		
NF5	$B \sqsubseteq C \sqcap D \rightarrow \{B \sqsubseteq C, B \sqsubseteq D\}$		

2. Proof of Proposition 1

Proof.

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

We prove $C \triangle D$ 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 $C \triangle D$ always exists.

We now prove unicity of $C \triangle D$ by contradiction. Suppose there are 2 concepts E_1 and E_2 , with $E_1 \neq E_2$ in $C \triangle D$. 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, $C \triangle D$ is unique.

b. Termination By induction, we show algorithm 1 always terminates and generates an output which size, we call s_{out} , is strictly less than 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 $\text{size}(D) \geq 1$).

- 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 $s_{in}^{rc,1} = \text{size}(C) + \text{size}(D_1)$.

$$\text{Thus: } s_{in}^{rc,1} \leq \text{size}(C) + n - \text{size}(C) - 2.m + 3 = n - 2.m + 3$$

$$\text{Thus: } s_{in}^{rc,1} \leq n - 1 \leq n \text{ (since } m \geq 2\text{).}$$

$$\text{By IH, the recursive call stops with } s_{out}^{rc,1} < s_{in}^{rc,1} \leq n - 2.m + 3.$$

- 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:

$$s_{in}^{rc,2} < \text{size}(s_{in}^{rc,1}) + \text{size}(D_2)$$

$$s_{in}^{rc,2} < \text{size}(s_{in}^{rc,1})$$

$$+ n + 1 - \text{size}(C) - \text{size}(D_1) - \text{size}(D_3) - \dots - \text{size}(D_m) - m + 1$$

$$s_{in}^{rc,2} \leq \text{size}(s_{in}^{rc,1})$$

$$+ n + 1 - \text{size}(C) - \text{size}(D_1) - \text{size}(D_3) - \dots - \text{size}(D_m) - m$$

$$s_{in}^{rc,2} \leq \text{size}(C) + \text{size}(D_1)$$

$$+ n + 1 - \text{size}(C) - \text{size}(D_1) - \text{size}(D_3) - \dots - \text{size}(D_m) - m$$

$$s_{in}^{rc,2} \leq n + 1 - \text{size}(D_3) - \dots - \text{size}(D_m) - m$$

$$s_{in}^{rc,2} \leq n + 1 - (m - 2) - m = n - 2.m + 3 \text{ (since } \text{size}(D_j) \geq 1, \forall 1 \leq j \leq m\text{)}$$

$$\text{By IH, the recursive call stops with } s_{out}^{rc,2} < s_{in}^{rc,2} \leq n - 2.m + 3.$$

- 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:

$$s_{in}^{rc,3} < \text{size}(s_{in}^{rc,2}) + \text{size}(D_3)$$

$$s_{in}^{rc,3} < \text{size}(s_{in}^{rc,2})$$

$$+ n + 1 - \text{size}(C) - \text{size}(D_1) - \text{size}(D_2) - \text{size}(D_4) - \dots - \text{size}(D_m) - m + 1$$

$$s_{in}^{rc,3} \leq \text{size}(s_{in}^{rc,2})$$

$$+ n + 1 - \text{size}(C) - \text{size}(D_1) - \text{size}(D_2) - \text{size}(D_4) - \dots - \text{size}(D_m) - m$$

$$s_{in}^{rc,3} \leq \text{size}(s_{out}^{rc,1}) + \text{size}(D_2)$$

$$+ n + 1 - \text{size}(C) - \text{size}(D_1) - \text{size}(D_2) - \text{size}(D_4) - \dots - \text{size}(D_m) - m$$

$$s_{in}^{rc,3} < \text{size}(s_{in}^{rc,1}) + \text{size}(D_2)$$

$$+ n + 1 - \text{size}(C) - \text{size}(D_1) - \text{size}(D_2) - \text{size}(D_4) - \dots - \text{size}(D_m) - m$$

$$s_{in}^{rc,3} < \text{size}(C) + \text{size}(D_1) + \text{size}(D_2)$$

$$+ n + 1 - \text{size}(C) - \text{size}(D_1) - \text{size}(D_2) - \text{size}(D_4) - \dots - \text{size}(D_m) - m$$

$$s_{in}^{rc,3} \leq \text{size}(C) + \text{size}(D_1) + \text{size}(D_2)$$

$$+ n + 1 - \text{size}(C) - \text{size}(D_1) - \text{size}(D_2) - \text{size}(D_4) - \dots - \text{size}(D_m) - m - 1$$

$$s_{in}^{rc,3} \leq n + 1 - \text{size}(D_4) - \dots - \text{size}(D_m) - m - 1$$

$$s_{in}^{rc,3} \leq n + 1 - (m - 2) - m = n - 2.m + 3 \text{ (since } \text{size}(D_j) \geq 1, \forall 1 \leq j \leq m\text{)}$$

$$\text{By IH, the recursive call stops with } s_{out}^{rc,3} < s_{in}^{rc,3} \leq n - 2.m + 3.$$

- 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 [4], 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 [4]. First let's define what is the description tree of a concept.

Definition 1 (description tree [4]). *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).*

[4] 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 2 (homomorphism of description trees [4]). *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 ([4]). 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 $\text{size}(D_1) \leq n - 1$ 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 φ 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:

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

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

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

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

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

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

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

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

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

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 , then 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 \square , 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 $\star\star \varphi$ 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 $\star\star$, 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:

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

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

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

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

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:

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

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

In the case where $\text{br} \neq \emptyset$, $\textcircled{2}$ implies $\text{br}_E \subseteq \text{br}_C$. Now corollary 1 says that: $\textcircled{1} C \sqsubseteq E$ 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 $\textcircled{1}$ and $\textcircled{2}$. So $E' = E$ and thus $E \in \text{subd}_C$. Since $E \in \text{subd}_C$ implies $C \sqsubseteq E$, we have:

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

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

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

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

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

Step 3

On the basis of the previous characterization of TSO, we now show $C \triangle D = \text{tso}(C, D)$ 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 $C \triangle D = \top$, 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 $C \triangle D = \top$, 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 $C \triangle D = \top$, 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 $C \triangle D = C$. 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 $s'_{in} = \text{size}(C') + \text{size}(D') \leq n_r$, $\text{tso}(C', D') = C' \triangle D'$.

- 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 $C \triangle D = \top$, 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 $C \triangle D = \top$, 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:

ⓑ $\forall i \in \{1, \dots, k\}$,
 $\text{tso}(C_i, D) = C_i \triangle D$

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

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 $E = \prod_{i=1}^k \text{tso}(C_i, D)$ since the output of the algorithm is built from it (modulo lines 7 and 9). Thus $\text{br}_E = \text{br}_{\prod_{i=1}^k \text{tso}(C_i, D)}$. According to ⑥, 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:

$$\text{br}_E = \text{br}_{\prod_{i=1}^k \text{tso}(C_i, D)} = \bigcup_{i=1}^k \text{br}_{\text{tso}(C_i, D)}$$

$$= \begin{cases} \bigcup_{i=1}^k \text{br}_i & \text{if } \bigcup_{i=1}^k \text{br}_i \neq \emptyset \\ \{\top\} & \text{if } \bigcup_{i=1}^k \text{br}_i = \emptyset \end{cases}$$

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

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

And thanks to ②, 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:

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

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

...

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

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

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 :

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

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

...

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

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

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

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

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. $\text{tso}(C, D) = E = C \triangle D$.

- 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:

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

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:

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

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 $\text{tso}(C, D) = C \triangle D$ 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 $C \triangle D = C$. 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: $C \triangle D$ (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 Result1 **then**
- 7: $\text{Result} := \text{Result1}$ without any \top conjunct. {Case 3}

```

8:   else
9:      $Result := \top$  {Case 4}
10:  end if
11:  else if  $D = D_1 \sqcap \dots \sqcap D_m$  with  $m \geq 2$  then
12:     $Result := \text{tso}(\dots(\text{tso}(\text{tso}(C, D_1), D_2), \dots), D_m)$  {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

$(Min(n_C, n_D) + 1) * k$ $+k$ $+p * k$ $+\sum_{i=1}^p T(size(C_i) + n_D)$ $+(p - 1) * k$ $+k$ $+p * k$ $+k,$	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 $C = C_1 \sqcap \dots \sqcap C_p$ L11: $D = D_1 \sqcap \dots \sqcap D_m$ 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 $C = C_1 \sqcap \dots \sqcap C_p$ L11: $D = D_1 \sqcap \dots \sqcap D_m$ 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 $C = C_1 \sqcap \dots \sqcap C_p$ L11: $D = D_1 \sqcap \dots \sqcap D_m$ 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

$$\begin{array}{ll}
(\text{Min}(n_C, n_D) + 1) * k & \text{L1: } C = D \text{ and } C = \top \\
+k & \text{L4: test } C = C_1 \sqcap \dots \sqcap C_p \\
+k & \text{L11: } D = D_1 \sqcap \dots \sqcap D_m \\
+2 * k & \text{L13: } C = \exists r.C' \text{ and } D = \exists r.D' \\
+k & \text{L21. variable assignment}
\end{array} \left. \vphantom{\begin{array}{l} \\ \\ \\ \\ \end{array}} \right\} \text{case 8}$$

After simplification, we get:

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

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:

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

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:

$$T(n) = (\text{Min}(n_C, n_D) + 2 + 4p) * k + p * (5 + 8m) * k$$

$$= k * \text{Min}(2p - 1, 2m - 1) + 4k * p + 2k + 5k * p + 8k * mp$$

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 $\mathcal{O}(n^2)$. □

References

- [1] F. Baader, S. Brandt, C. Lutz, Pushing the \mathcal{EL} envelope, in: Proceedings of the 19th International Joint Conference on Artificial Intelligence, IJCAI'05, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005, p. 364–369.
- [2] R. Peñaloza, A.-Y. Turhan, A practical approach for computing generalization inferences in el, in: The Semantic Web: Research and Applications, Springer Berlin Heidelberg, 2011, pp. 410–423.
- [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] 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>.