Reaching for the stars in $\mathcal{EL}^*$ concept learning

Bente Gortworst[1], Cem Okulmus[2], Magdalena Ortiz[1] and Anni-Yasmin Turhan[2]

[1]TU Wien, Vienna, Austria
[2]Paderborn University, Paderborn, Germany

Abstract

Learning concepts from data examples is an intensively investigated topic in description logics (DLs), in particular for the $\mathcal{EL}$-family. So far, hardly any of these works on concept learning consider the Kleene star to express the (reflexive) transitive closure of roles. However, the star plays a central role in some closely related settings, like learning shape expressions in graph constraint languages such as SHACL, and learning graph queries from graph data. As a first step towards developing techniques for learning SHACL shape expressions, we consider learning from examples (aka the fitting problem) for the logic $\mathcal{EL}^*$, which extends $\mathcal{EL}$ by the reflexive transitive closure of named roles. In contrast to $\mathcal{EL}$, the DL $\mathcal{EL}^*$ can express a limited form of disjunction that can result in more informative fittings. By adapting the automata-based techniques for $\mathcal{EL}$, we obtain complexity bounds for deciding existence of $\mathcal{EL}^*$ fittings that coincide with those for $\mathcal{EL}$. We also develop an algorithm for computing most specific fittings in $\mathcal{EL}^*$.

Keywords

Concept learning, EL family, fitting problem, learning from examples

1. Introduction

Learning formulas from data examples has long been a central problem in AI, and it is currently receiving renewed attention. In the context of description logics (DLs), various settings for such concept learning have been studied under different names. Early works considered learning concepts from positive examples only [1, 2, 3, 4] and investigated methods for computing what are nowadays called most-specific fittings. The last decade has seen a lot of contributions from the DL and the data base theory community regarding learning from examples, where additionally negative examples are provided and the goal is to compute a *fitting*, i.e., a complex formula that captures the commonalities of the positive examples and separates these from the negative ones. In particular, the problem of learning DL concepts from examples has been investigated in the presence of TBoxes from interpretations [5, 6] or even from ABoxes [4, 7]. This problem is well understood with elegant algorithms, well-characterized computational complexity, and even practical systems [7, 8]. Comparable progress has been achieved for the problem of learning conjunctive queries (CQs) from data bases [9, 10, 11], and there has also been interest in learning linear temporal logic (LTL) formulas from temporal data [12, 13].

A similar interest in learning complex expressions from data examples has been manifesting in the *Shapes Constraint Language* [14] (SHACL) community. The W3C recommendation SHACL is a language for the validation of RDF graphs. It allows us to express constraints on RDF graphs by defining complex *shapes* that can be validated at target nodes of a graph. There are enormous amounts of RDF datasets and many of them have been created over decades without any schema guiding their population. It is thus not surprising that learning SHACL shapes from data examples has been considered in a few works [15, 16, 17]. However, all of these approaches thusfar are pragmatic and focus on mining rather simple shapes. To our knowledge, there have been no systematic attempts at understanding the properties and limits of the shape learning algorithms.

As it turns out, SHACL shapes are essentially concepts in an expressive DL extending $\mathcal{ALCOIQ}^{reg}$ [18, 19], and hence learning DL concepts can be viewed as learning shapes in a SHACL fragment.

DL 2026: 39th International Workshop on Description Logics, July 17–19, 2026, Lisbon, Portugal
bente.gortworst@tuwien.ac.at (B. Gortworst); cem.okulmus@upb.de (C. Okulmus); ortiz@kr.tuwien.ac.at (M. Ortiz); turhan@uni-paderborn.de (A. Turhan)
0000-0002-7742-0439 (C. Okulmus); 0000-0002-2344-9658 (M. Ortiz); 0000-0001-6336-335X (A. Turhan)

This facilitates the transfer of results on learning problems and their properties from the DL to the SHACL setting. However, the DLs for which concept learning has been studied lack some features that are central to SHACL. In particular, they do not support reachability or complex path expressions for describing paths of varying lengths, which are central when constraining RDF graphs.

In this paper, as a first step towards understanding concept learning in the presence of path expressions, we study the concept learning problem for the language $\mathcal{EL}^*$, whose concepts correspond to shapes in a non-trivial SHACL fragment and that extend $\mathcal{EL}$ by the use of the Kleene star in existential restrictions. Specifically, we consider the setting where data is given as a finite edge- and node-labelled graph, where some nodes are labelled as positive and some as negative examples, and without a TBox. The goal is to find a fitting concept in $\mathcal{EL}^*$ that is satisfied at all positive nodes and at none of the negative ones. Interestingly, the use of r^* is effectively a disjunction over r-paths of differing length and as such adds to the expressivity of the target language. Consider the graph below, where green nodes are positive examples:

While their commonalities expressed in $\mathcal{EL}$ yields $\exists r.\top$, in $\mathcal{EL}^*$ it gives $(\exists r^*.A) \sqcap (\exists r^*.B)$, witnessing the effect of the increased expressivity of $\mathcal{EL}^*$.

As fittings need not always exist, we investigate the problem of deciding existence of fittings. For $\mathcal{EL}$ concepts, the existence of fittings is known to be ExpTime-complete already in the absence of a TBox and negative examples [1]. We show here that the increased expressivity of r^*-paths does not change the complexity. To attain this result we could not merely extend the results on $\mathcal{EL}^+$ from [2], since there only tree-shaped input is used, while our input graph may well be cyclic. Instead, we adapted the approach based on alternating parity tree automata from [11] to r^*-paths.

We also study most specific fittings (MSFs) expressed in $\mathcal{EL}^*$ and give procedures for deciding existence and for computing the MSF, if it exists. The standard approach for such computation algorithms is to use a cross-product construction of the graphs reachable from positive examples and test whether the resulting construction excludes the negative examples. Our procedure for $\mathcal{EL}^*$ adapts this approach by first extending the graph by the reflexive, transitive closure of the r-reachability. We develop a new notion of simulation to detect if this new product fits the examples. Finally, to obtain a finite shape from the product graph, the latter needs to be unravelled. We present an algorithm to compute the MSF and show that, if the number of positive examples is bounded, deciding fitting existence and computing a succinct representation of the MSF can even be done in polynomial time.
Some of the proofs for our results can be found in the appendix.

2. Preliminaries

We consider disjoint, countably infinite sets of concept names $\mathsf{N_C}$ and roles names $\mathsf{N_R}$. With $A \in \mathsf{N_C}$ and $r \in \mathsf{N_R}$, $\mathcal{EL}^*$ *concepts* are defined by the following grammar:

$$C \to \top \mid A \mid \exists R.C \mid C \sqcap C \qquad\qquad R \to r \mid r^*$$

$\mathcal{EL}$ is the fragment of $\mathcal{EL}^*$ that disallows r^*.

The semantics of $\mathcal{EL}^*$ concepts is defined in terms of interpretations in the usual way, but we use a different notation: *(interpretation) graphs* are triples $\mathcal{G} = (V, E, \ell)$; The set V of nodes is the *domain* of $\mathcal{G}$, while $E \subseteq V \times \mathsf{N_R} \times V$ is the labelled *edge* relation; we may write $(u, v) \in r^{\mathcal{G}}$ to indicate that $(u, r, v) \in E$, and we define

$$(r^*)^{\mathcal{G}} = \{(v, v) \mid v \in V\} \cup \{(u, w) \mid \exists v_0, \cdots, v_n \in V.(v_0 = u) \wedge (w = v_n) \wedge (v_i, v_{i+1}) \in r^{\mathcal{G}} \text{ for } 0 \leq i \leq (n-1)\}$$

The node-labelling function $\ell : V \to 2^{\mathsf{N_C}}$ determines the interpretation of concept names. We may write $u \in A^{\mathcal{G}}$ to express that $A \in \ell(u)$, and extend this function to all $\mathcal{EL}^*$ concepts as expected:

$$(\exists R.C)^{\mathcal{G}} = \{u \mid (u, v) \in R^{\mathcal{G}} \text{ and } v \in C^{\mathcal{G}}\} \qquad (C_1 \sqcap C_2)^{\mathcal{G}} = C_1^{\mathcal{G}} \cap C_2^{\mathcal{G}} \qquad \top^{\mathcal{G}} = V.$$

$\mathcal{G}$ is *finite* if its sets of nodes and labels are; in this paper we consider finite graphs. An interpretation graph $\mathcal{G}$ is a *model* of a concept C, written $\mathcal{G} \models C$, if $C^{\mathcal{G}} \neq \emptyset$. We may write $\mathcal{G} \models C(u)$ whenever $u \in C^{\mathcal{G}}$. Given concepts C and D, concept D *subsumes* C (denoted $C \sqsubseteq D$), if in all graphs $\mathcal{G}$, it holds that $C^{\mathcal{G}} \subseteq D^{\mathcal{G}}$. A *pointed graph* is a pair $(\mathcal{G}, v)$ of a graph $\mathcal{G}$ and a node v in its domain.

A *signature* $\Sigma \subseteq \mathsf{N_R} \cup \mathsf{N_C}$ is a finite set of role and concept names. If an interpretation graph $\mathcal{G}$ is finite, we define its signature $\Sigma(\mathcal{G})$ as $\Sigma(\mathcal{G}) = \{r \mid r \in \mathsf{N_R} \text{ and } r^{\mathcal{G}} \neq \emptyset\} \cup \{A \mid A \in \mathsf{N_C} \text{ and } A^{\mathcal{G}} \neq \emptyset\}$. The signature of an object X, such as a complex concept or a labelled graph, $\Sigma(X)$ is the set of role and concept names that occur in X. We define a Σ-concept to be some concept whose signature is Σ. Given a signature Σ, we may write $\Sigma_{\mathsf{C}} = \Sigma \cap \mathsf{N_C}$ and $\Sigma_{\mathsf{R}} = \Sigma \cap \mathsf{N_R}$ to refer to its subsets. We let $\mathcal{G}|_\Sigma$ denote the restriction of a graph $\mathcal{G}$ to the signature Σ.

3. Fitting Problems for $\mathcal{EL}^*$

We study the following fitting problem: for a given graph $\mathcal{G}$ where some nodes may be marked as positive or negative examples, the goal is to compute a (complex) concept C that fits all the examples, that is, that it is satisfied at the positive nodes but not at the negative ones.

Definition 1 ($\mathcal{L}$-fitting)**.** *A* fitting instance *is a tuple $\mathcal{F} = (\mathcal{G}, \mathscr{P}, \mathscr{N}, \Sigma)$, where $\mathcal{G}$ is a finite labelled graph, $\mathscr{P}$ and $\mathscr{N}$ are the* positive *and* negative *nodes, respectively, such that $\mathscr{P} \cup \mathscr{N} \subseteq V$. Σ is a signature, where $\Sigma \subseteq \Sigma(\mathcal{G})$. For a concept language $\mathcal{L}$, an $\mathcal{L}$* fitting *for $\mathcal{F}$ is an $\mathcal{L}$ concept C with $\Sigma(C) \subseteq \Sigma$ such that $\mathcal{G} \models C(p)$ for every $p \in \mathscr{P}$ and $\mathcal{G} \not\models C(n)$ for every $n \in \mathscr{N}$.*

We may omit the signature if $\Sigma = \Sigma(\mathcal{G})$ and write simply $\mathcal{F} = (\mathcal{G}, \mathscr{P}, \mathscr{N})$. Depending on the fitting instance, there may exist no fitting, or there may be many. In the latter case, the most informative fittings are often preferred. These are formalised in terms of subsumption as follows.

Definition 2 (Most specific $\mathcal{L}$ fitting)**.** *Given a fitting instance $\mathcal{F} = (\mathcal{G}, \mathscr{P}, \mathscr{N}, \Sigma)$ a* most specific $\mathcal{L}$-fitting *($\mathcal{L}$ MSF) C is an $\mathcal{L}$ fitting for $\mathcal{F}$ and for every $\mathcal{L}$ fitting C' for $\mathcal{F}$ we have $C \sqsubseteq C'$.*

$\mathcal{EL}^*$ concepts can be conveniently represented as trees, analogous to the $\mathcal{EL}$ case [20]. In such trees, we use special symbols $r_\star$ to represent existential concepts using the Kleene star. We let $\mathsf{N}^\star_{\mathsf{R}} = \mathsf{N_R} \cup \{r_\star \mid r \in \mathsf{N_R}\}$. For a signature Σ, we define its *extended role signature* $\Sigma^\star_{\mathsf{R}} = \Sigma_{\mathsf{R}} \cup \{r_\star | r \in \Sigma_{\mathsf{R}}\}$, and we use $\Sigma^\star_{\mathsf{R}}(X) = \{r, r_\star \mid r \in \Sigma_{\mathsf{R}}(X)\}$ to denote the extended role signature of a finite object X.

Definition 3 ($\mathcal{EL}^*$ Description Tree)**.** *An $\mathcal{EL}^*$* description tree *is a finite edge- and node-labelled tree-shaped graph $\mathcal{T} = (V, E, \ell)$ with $E \subseteq V \times \mathsf{N}^\star_{\mathsf{R}} \times V$ and $\ell : V \to 2^{\mathsf{N_C}}$. Let C be an $\mathcal{EL}^*$ concept in the form $A_0 \sqcap \cdots \sqcap A_n \sqcap \exists R_1.C_1 \sqcap \cdots \sqcap \exists R_m.C_m$ with $A_i \in \mathsf{N_C}$, where $0 \leq i \leq n$ and each C_j is an $\mathcal{EL}^*$ concept and $R_j \in \{r, r^* \mid r \in \mathsf{N_R}\}$, with $1 \leq j \leq m$. Given a concept C in this form, its* role depth *$depth(C)$ is defined as usual, as the maximal depth of its quantifiers. C can be translated into its $\mathcal{EL}^*$* description tree *$\mathcal{T}_C = (V, E, \ell)$, defined by an induction on the role depth of C as follows. Let u_0 be the root of $\mathcal{T}_C$. If $depth(C) = 0$, then $V = \{u_0\}$, $E = \emptyset$ and $\ell(u_0) = \{A_0, \ldots, A_n\}$.*
If $depth(C) > 0$, for $1 \leq j \leq m$, let $\mathcal{T}_{C_j} = (V_j, E_j, \ell_j)$ be the inductively defined description tree corresponding to C_j and u_j be its root, where we assume that $V_1, \ldots V_n$ are pairwise disjoint, then

$$V = \{u_0\} \cup \bigcup_{1 \leq j \leq m} V_j \qquad \ell(v) = \begin{cases} \{A_0, \ldots, A_n\} & v = u_0 \\ \ell_j(v) & v \in V_j, 1 \leq j \leq m \end{cases}$$

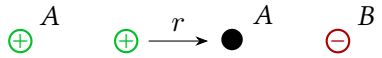
$$f(R) = \begin{cases} r_\star & \text{if } R = r^* \text{ and } r \in \mathsf{N_R} \\ R & \text{otherwise} \end{cases} \qquad E_C = \bigcup_{1 \leq j \leq m} \{u_0, f(R_j), u_j\} \cup \bigcup_{1 \leq j \leq m} E_j$$

We call C the *corresponding concept* of $\mathcal{T}_C$. We sometimes consider the *pointed* graph $(\mathcal{T}, u_0)$ that pairs a $\mathcal{T}$ with its root u_0; slightly abusing notation, we denote this pointed graph as simply $\mathcal{T}$.

4. Complexity of Fitting Existence

We study the complexity of deciding the existence of an $\mathcal{EL}^*$-fitting for an instance $\mathcal{F} = (\mathcal{G}, \mathscr{P}, \mathscr{N}, \Sigma)$, and show that the problem is ExpTime-complete. Deciding $\mathcal{EL}$ fitting existence is well-known to be hard for exponential time [7], but this does not necessarily transfer to $\mathcal{EL}^*$, since more expressive fitting languages generally admit more fittings. While the existence of an $\mathcal{EL}$ fitting trivially implies the existence of an $\mathcal{EL}^*$ fitting, the converse does not hold.

Example 1. *Consider $\mathcal{F} = (\mathcal{G}, \mathscr{P}, \mathscr{N}, \Sigma)$ with $\mathscr{P}$ marked via green nodes and $\mathscr{N}$ marked red below. $\mathcal{F}$ has no $\mathcal{EL}$ fittings, but $\exists r^*.A$ is an $\mathcal{EL}^*$ fitting.*



Despite the increased expressivity, the $\mathcal{EL}^*$-fitting existence problem is not easier in the worst case. The hardness holds even without signature restrictions.

Proposition 1. *Deciding the existence of an $\mathcal{EL}^*$-fitting for a given $\mathcal{F} = (\mathcal{G}, \mathscr{P}, \mathscr{N})$ is ExpTime-hard.*

Proof sketch. To show that fitting existence is ExpTime-hard, we reduce from the word problem for an alternating, linear space bounded Turing machine M. We reuse the reduction by Funk et al. (2019) (Theorem 12) for the $\mathcal{EL}$-fitting existence problem. We build from M and an input word w, an instance $\mathcal{F}_M = (\mathcal{G}_M, \mathscr{P}_w, \{n\})$. $\mathcal{G}_M$ is the union of two graphs. In a nutshell, the nodes of the first graph represent possible configurations of M restricted to a single tape position, and their connections reflect the local effects of executing the transitions of M. The second graph represents how acceptance propagates along existential and universal states in a run of M. The positive examples represent the initial tape contents in such a way that possible fittings represent correct executions of M on w, while the single negative example restricts the fittings to non-accepting runs. The authors show that there exists an $\mathcal{EL}$-fitting for $\mathcal{F}_M$ iff there is no accepting run of M on w. To lift this claim from $\mathcal{EL}$ to $\mathcal{EL}^*$, we leverage a property of $\mathcal{G}_M$: by construction, all paths in $\mathcal{G}_M$ that start at some node in $\mathscr{P}_w$ move along edges whose labels strictly alternate between (roles representing) existential and universal states of M, until an accepting or rejecting state is reached. Because of this strict alternation, r^*-edges can only represent r-paths of length exactly one, and hence it is possible to show that for every $\mathcal{EL}^*$-fitting for $\mathcal{F}_M$, one can define a corresponding $\mathcal{EL}$-fitting for $\mathcal{F}_M$ by replacing any r^*-edges by regular r-edges. Once this has been established, we can argue as in Funk et al. (2019) that there exists an $\mathcal{EL}^*$-fitting C for $\mathcal{F}_M$ iff M does not accept w. □

To characterize fittings we use pre-matches, which are homomorphisms from trees to pointed graphs.

Definition 4. *A* pre-match *for a tree $\mathcal{T} = (V, E, \ell)$ with root v_0 into a pointed graph $(\mathcal{G}, u)$, is a function $\rho : V \to \mathcal{G}$ such that $\rho(v_0) = u$ and for all $v, v' \in V$:*

1. *$\rho(v) \in A^{\mathcal{G}}$ for every $A \in \ell(v)$,*
2. *if $(v, r, v') \in E$ then $(\rho(v), \rho(v')) \in r^{\mathcal{G}}$, and*
3. *if $(v, r_\star, v') \in E$ then $(\rho(v), \rho(v')) \in (r^*)^{\mathcal{G}}$.*

Lemma 1. *For every $\mathcal{EL}^*$ concept C, there is a pre-match ρ from its $\mathcal{EL}^*$ description tree $\mathcal{T}_C$ into $(\mathcal{G}, u)$ iff $\mathcal{G} \models C(u)$, where $u = \rho(v_0)$ and v_0 is the root of $\mathcal{T}_C$.*

We present an algorithm for fitting existence that uses tree automata to decide the existence of suitable pre-matches. Since the automata run on (strictly) node-labelled full k-ary trees, we introduce an alternative representation of our description trees.

Definition 5 (Node-labelled $\mathcal{EL}^*$ k-Trees)**.** *We define the alphabet $\mathbb{L}_\Sigma = (\{-\} \cup \mathsf{N}_\mathsf{R}^\star) \times 2^{\Sigma_\mathsf{C}}$. For an element $w \in \mathbb{L}_\Sigma$ with $w = (r, \mathbf{A})$, we access its components by $w.R = r$ and $w.\mathbf{A} = \mathbf{A}$.*

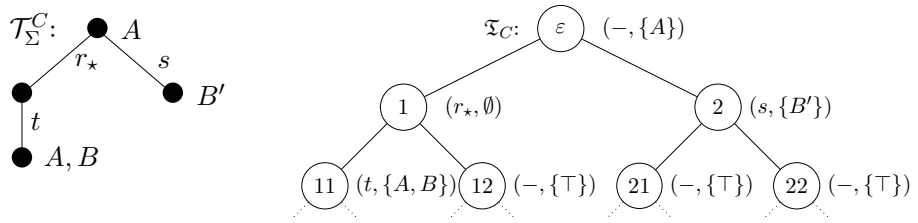


Figure 1: The $\mathcal{EL}^*$ tree $\mathcal{T}_C$ and the node-labelled $\mathcal{EL}^*$ 2-tree $\mathfrak{T}_C$ of the $\mathcal{EL}^*$ concept $C = A \sqcap \exists r^*.\exists t.(A \sqcap B) \sqcap \exists s.B'$.

For $k \geq 0$, a node-labelled $\mathcal{EL}^*$ k-tree $\mathfrak{T} = (V, \ell)$ *has the set V is the set of all words over the alphabet $\{1, \ldots, k\}$ as nodes, and a labelling function $\ell : V \to \mathbb{L}_\Sigma$. We call such a tree* proper *if the root ε has $\ell(\varepsilon) = (-, \mathbf{A})$ for some $\mathbf{A} \neq \{\top\}$, and for each $w \neq \varepsilon$ with $w.R = -$, the labelling is such that $\ell(w) = (-, \{\top\})$ for every $w \in V$, and every branch in $\mathfrak{T}$ reaches a node labelled $(-, \{\top\})$ in a finite number of steps. If the labels mention only symbols in a signature Σ, we may say that $\mathfrak{T}$ is over Σ.*

Given an $\mathcal{EL}^*$ description tree $\mathcal{T} = (V, E, \ell)$ and a $k \geq deg(\mathcal{T})$, let n be an arbitrary but fixed function that assigns to each node in V a word over $\{1, \ldots, k\}$ in such a way that: (i) $n(v_0) = \varepsilon$ for the root v_0 of $\mathcal{T}$, and (ii) $n(v_i) = n(v)i$ for the children $v_1, \ldots, v_n$ of v. We define the encoding $\mathsf{nlt}_k(\mathcal{T})$ of $\mathcal{T}$ as the node-labelled k-tree $\mathfrak{T} = (V', \ell_\mathcal{T})$ that has:

$$\begin{aligned}
\ell_\mathcal{T}(\varepsilon) &= (-, \ell(v_0)) && \text{for the root } v_0 \text{ of } \mathcal{T} \\
\ell_\mathcal{T}(wj) &= (r, \ell(v_j)) && \text{for each } wj = n(v_j) \text{ with } (n(w), r, v_j) \in E \\
\ell_\mathcal{T}(w) &= (-, \{\top\}) && \text{whenever there is no } v \in V \text{ with } n(v) = w
\end{aligned}$$

As each proper node-labelled $\mathcal{EL}^*$ k-tree $\mathfrak{T}$ is the encoding $\mathsf{nlt}_k(\mathcal{T})$ of some $\mathcal{EL}^*$ description tree $\mathcal{T}$, we may say that $\mathfrak{T}$ has a pre-match into a pointed graph if $\mathcal{T}$ does. If $\mathfrak{T}$ is the corresponding node-labelled $\mathcal{EL}^*$ k-tree for some $\mathcal{T}_C$, we write $\mathfrak{T}_C$ and we refer to C as its corresponding shape.

Figure 1 illustrates the two tree notions $\mathcal{T}_C$ and $\mathfrak{T}_C$ for an $\mathcal{EL}^*$ concept C.

Alternating parity tree automata. We recall the basics of alternating automata on infinite node-labelled k-trees. For the presentation, we largely follow [21] and [11]. Let $\mathcal{B}(I)$ be the set of positive Boolean formulae over a set of variables I, built inductively by applying $\wedge$ and $\vee$ starting from $\bot$, $\top$ and the elements of I. For a set $J \subseteq I$ and a formula $C \in \mathcal{B}(I)$, we say J *satisfies* C iff assigning $\top$ to the elements in J and $\bot$ to those in $I \setminus J$, makes C true. For a positive integer k, let $[k] = \{0, 1, \ldots, k\}$. A *$k$-ary alternating parity tree automaton* (k-ATA) running over infinite node-labelled k-trees, is a tuple $\mathfrak{A} = \langle \Sigma, Q, \delta, q_0, p \rangle$, where Σ is an input alphabet, Q is a finite set of states, $\delta : Q \times \Sigma \to \mathcal{B}([k] \times Q)$ is the transition function, $q_0 \in Q$ is the initial state and $p : Q \to \mathbb{N}$ is a priority function.

Deciding fitting existence with ATAs. Our goal now is to build a k-ATA $\mathfrak{A}_\mathcal{F}$ that accepts exactly the proper node-labelled $\mathcal{EL}^*$ k-trees that represent a fitting for a given fitting instance $\mathcal{F}$. We first define, for a pointed graph $(\mathcal{G}, v_0)$ and $k \geq 0$, a k-ATA $\mathfrak{A}^{\mathcal{G}, v_0}$, which takes as input proper node-labelled $\mathcal{EL}^*$ k-trees and accepts exactly those with a pre-match into $(\mathcal{G}, v_0)$.

Let $\mathcal{G} = (V, E, \ell)$ and $v_0 \in V$. The k-ATA $\mathfrak{A}^{\mathcal{G}, v_0} = \langle Q, \mathbb{L}_\Sigma, \delta, q_{v_0}, p \rangle$ has the state set Q and transition function $\delta : Q \times \mathbb{L}_\Sigma \to \mathcal{B}([k] \times Q)$ defined as below, and the priority function p has $p(q_v) = 0$ for the states of the form q_v with $v \in V$, and $p(q) = 1$ for all other states.

$$Q = \{q_v \mid v \in V\} \cup q_- \cup \{q_r \mid r \in \Sigma_\mathsf{R}(\mathcal{G})\} \cup \{q_{r^*} \mid r \in \Sigma_\mathsf{R}(\mathcal{G})\} \cup \{q_{r^* v} \mid v \in V, r \in \Sigma_\mathsf{R}(\mathcal{G})\}$$

$$\delta(q_v, w) = \begin{cases} \displaystyle\bigwedge_{1 \leq i \leq k} \Bigg((i, q_-) \vee \Big((i, q_{r^*}) \wedge (i, q_v) \Big) \vee \\ \qquad \displaystyle\bigvee_{r(v,v') \in E} \Big(\big((i, q_r) \wedge (i, q_{v'}) \big) \vee \big((i, q_{r^*}) \wedge (i, q_{r^* v'}) \big) \Big) \Bigg) & \text{if } v \in A^\mathcal{G} \\ & \text{for every } A \in w.\mathbf{A} \\ \bot & \text{otherwise} \end{cases}$$

$$\delta(q_{r^*v}, w) = (0, q_v) \vee \bigvee_{r(v,v') \in E} (0, q_{r^*v'})$$

$$\delta(q_-, w) = \begin{cases} \top & \text{if } - = w.R \\ \bot & \text{otherwise} \end{cases} \qquad \delta(q_r, w) = \begin{cases} \top & \text{if } r = w.R \\ \bot & \text{otherwise} \end{cases} \qquad \delta(q_{r^*}, w) = \begin{cases} \top & \text{if } r_\star = w.R \\ \bot & \text{otherwise} \end{cases}$$

Intuitively, a succesful run of $\mathfrak{A}^{\mathcal{G},v_0}$ can visit a node w of $\mathfrak{T}$ in state q_v if there is a pre-match ρ from $\mathfrak{T}$ into v such that $\rho(w) = v$. Similarly, a visit to w on state q_{r^*v} will succeed if there is a pre-match ρ from $\mathfrak{T}$ into $\mathcal{G}$, but $\rho(w) = v$ is not enforced: it suffices if the node $\rho(w)$ is reachable from v along a path of r-edges. The remaining states only check for the correct role labels. All transitions except those involving q_{r^*v} require the automaton to move to a new node in $\mathfrak{T}$, and the automaton halts whenever it reaches a node labelled $(-, \{\top\})$. This, together with the priority condition, ensures that all accepting runs on proper node-labelled k-trees are finite and witness the existence of a pre-match for the entire concept represented in $\mathfrak{T}$. The following lemma establishes the correctness of the automaton.

Lemma 2. *Given a pointed graph $(\mathcal{G}, v_0)$, the k-ATA $\mathfrak{A}^{\mathcal{G},v_0}$ accepts a proper node-labelled $\mathcal{EL}^*$ k-tree $\mathfrak{T}$ iff it has a pre-match into $(\mathcal{G}, v_0)$.*

Given $\mathfrak{A}$, we denote by $\overline{\mathfrak{A}}$ its *complement automaton*. We also denote by $\mathfrak{A}_{\mathscr{A}_n}$ the intersection automaton over a set of k-ATAs $\mathscr{A}_n = \mathfrak{A}_1, \ldots, \mathfrak{A}_n$. Both complementation and intersection are trivial for ATAs, please see the appendix for details. If we take the complement automaton $\overline{\mathfrak{A}^{\mathcal{G},v_0}}$ of the k-ATA $\mathfrak{A}^{\mathcal{G},v_0}$, it will accept exactly the proper node-labelled k-trees that do not have a pre-match into $(\mathcal{G}, v_0)$. Hence, for a fitting instance $\mathcal{F} = (\mathcal{G}, \mathscr{P}, \mathscr{N}, \Sigma)$, we can take the intersection $\mathfrak{A}_{\mathscr{A}_m}$ of the set of automata $\mathscr{A}_m$ that contains $\mathfrak{A}^{\mathcal{G},v}$ for every $v \in \mathscr{P}$ and $\overline{\mathfrak{A}^{\mathcal{G},n}}$ for each $n \in \mathscr{N}$. This automaton will accept exactly those proper node-labelled k-trees over the signature Σ that represent fittings for $\mathcal{F}$.

To decide fitting existence, we still have to show that we can pick a sufficiently large k so that the intersection automata will always accept some fitting if it exists. This is done in the following lemma.

Lemma 3. *Given an instance $\mathcal{F} = (\mathcal{G}, \mathscr{P}, \mathscr{N}, \Sigma)$, where $\mathcal{G} = (V, E, \ell)$, if there is an $\mathcal{EL}^*$ fitting for $\mathcal{F}$, then there is an $\mathcal{EL}^*$ fitting C such that $deg(\mathcal{T}_C) = |V|$.*

We now have an automaton that accepts the tree encoding of a fitting whenever it exists.

Theorem 1. *Given a fitting instance $\mathcal{F} = (\mathcal{G}, \mathscr{P}, \mathscr{N}, \Sigma)$, let k be the number of nodes in $\mathcal{G}$, and let $\mathfrak{A}_\mathcal{F}$ be the k-ATA obtained by intersecting $\mathfrak{A}^{\mathcal{G},v}$ for every $v \in \mathscr{P}$ and $\overline{\mathfrak{A}^{\mathcal{G},n}}$ for each $n \in \mathscr{N}$. Then $\mathfrak{A}_\mathcal{F}$ accepts a proper node-labelled k-tree $\mathfrak{T}_C$ over the signature Σ if and only if C is an $\mathcal{EL}^*$ fitting for $\mathcal{F}$.*

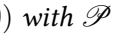
Finally, an emptiness check on $\mathfrak{A}_\mathcal{F}$, which can be performed in ExpTime, then gives us the ExpTime-upper bounds on the $\mathcal{EL}^*$ fitting existence problem.

Theorem 2. *The $\mathcal{EL}^*$ fitting existence problem is ExpTime-complete.*

Proof sketch. The hardness was shown in Proposition 1. Using Theorem 1, we can decide if an $\mathcal{EL}^*$ fitting exists for $\mathcal{F} = (\mathcal{G}, \mathscr{P}, \mathscr{N}, \Sigma)$ relying on a nonemptiness check on $\mathfrak{A}_\mathcal{F}$. Emptiness of ATAs is decidable in single exponential time in the combined size of the state set and the parity condition [22]; both are polynomially bounded in $\mathcal{F}$, so the test is feasible in time exponential in the size of $\mathcal{F}$. □

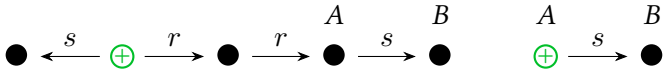
5. Computation of Most Specific Fittings

Before presenting our algorithms for the computation of most specific fittings, we note that, similarly to the case of arbitrary $\mathcal{EL}$ fittings, existence of $\mathcal{EL}^*$ fittings and of most specific $\mathcal{EL}^*$ fittings do not coincide. This is shown in Example 2, which is taken from [11], where it illustrates that the existence of $\mathcal{EL}$ fittings does not imply the existence of an $\mathcal{EL}$ MSF; by moving from $\mathcal{EL}$ to $\mathcal{EL}^*$ we obtain more fittings, but there still need not exist an MSF.

Example 2. *Consider $\mathcal{F} = (\mathcal{G}, \mathscr{P}, \emptyset)$ with $\mathscr{P}$ marked green in the graph:*  *. Then $C = \exists r^*.\top$ is an $\mathcal{EL}^*$ fitting for $\mathcal{F}$, while $D = \exists r.\top$ is both an $\mathcal{EL}$ and an $\mathcal{EL}^*$ fitting. As $D \sqsubseteq C$ holds, C is not an MSF. But D can be expanded into a more specific fitting $\exists r.(\top \sqcap \exists r.\top)$. Any fitting for $\mathcal{F}$ can be expanded similarly by further unravelling the graph. As a more specific fitting always exists, no fitting is an MSF.*

Even when MSFs exist for both $\mathcal{EL}$ and $\mathcal{EL}^*$, they need not coincide, as illustrated next.

Example 3. *Consider a fitting instance $\mathcal{F} = (\mathcal{G}, \mathscr{P}, \emptyset)$ with $\mathscr{P}$ marked green:*



The $\mathcal{EL}$ MSF is the concept $C_1 = \exists s.\top$. However, in $\mathcal{EL}^*$ we can express the concept $C_2 = \exists s.\top \sqcap \exists r^*.(A \sqcap \exists s.B)$, with $C_2 \sqsubseteq C_1$. In fact, C_2 is the $\mathcal{EL}^*$ MSF.

We will show that, even though an $\mathcal{EL}^*$ and $\mathcal{EL}$ MSF need not coincide, the existence of the former implies the existence of the latter. Interestingly, the converse fails, as illustrated by Example 4. We will return to this later in the section to make the claim more concrete.

Example 4. *Consider a fitting instance $\mathcal{F} = (\mathcal{G}, \mathscr{P}, \emptyset)$ with $\mathscr{P} = \{v_1, v_2\}$ marked green:*

There exists an $\mathcal{EL}$ MSF for $\mathcal{F}$, namely the concept A. But there is no $\mathcal{EL}^*$ MSF: while there are many $\mathcal{EL}^*$ fittings, none of these are most specific. For example, the fitting $A \sqcap \exists r^*.(\exists s^*.B)$ is less specific than $A \sqcap \exists r^*.(\exists s^*.(B \sqcap \exists r^*.(\exists s^*.A)))$. By traversing the cycle one more time, any $\mathcal{EL}^*$ concept can be expanded into a more specific one.

In the remainder of this section, we provide a computation algorithm for the $\mathcal{EL}^*$ MSF, if it exists.

For MSFs we do not use tree automata, but instead rely on the well-known notion of graph products, similarly to previous literature on MSFs [11, 7]. This more constructive approach allows us to not only decide whether an MSF exists, but also compute a representation of it.

To preserve $\mathcal{EL}^*$ concepts when doing graph product operations, we need to keep track of r^*-paths. We do this using the $r_\star$-symbols, similarly as we did for description trees. In the sequel, we call a graph *starred* if its signature has $r_\star$-symbols from $\Sigma_{\mathsf{R}}^\star$, and extend the interpretation function to take into account these symbols, such that $(r \cup r_\star)^{\mathcal{G}} = r^{\mathcal{G}} \cup (r_\star)^{\mathcal{G}}$ and $((r \cup r_\star)^*)^{\mathcal{G}}$ is its reflexive transitive closure, defined analogously to $(r^*)^{\mathcal{G}}$ in Section 2.

Graphs that witness all (non-trivial) r^*-paths by explicit $r_\star$-symbols are called $\star$-closed.

Definition 6. *Given a starred graph $\mathcal{G} = (V, E, \ell)$, we say that $\mathcal{G}$ is* $\star$-closed *if whenever there are nodes $v, v \in V$ with $v \neq v'$ and $(v, v') \in ((r \cup r_\star)^*)^{\mathcal{G}}$ and $(v, v') \notin r^{\mathcal{G}}$, we have $(v, v') \in r_\star^{\mathcal{G}}$.*

We can add $r_\star$-symbols to a given graph and make it $\star$-closed. We also define a simple 'clean-up' operation that removes from a starred graph some $r_\star$-symbols that are not necessary to be $\star$-closed.

Definition 7. *Given $\mathcal{G}$, let $\mathcal{G}_\star$ be the starred graph obtained by adding an edge $(u_0, r_\star, u_n)$ for all pairs (u_0, u_n) of nodes with u_n reachable from u_0 by a (possibly empty) sequence of r-edges $(u_0, r, u_1), \cdots, (u_{n-1}, r, u_n)$.*

The redundant star removal *of a given starred graph $\mathcal{G}$, denoted $\mathsf{rsr}(\mathcal{G})$, is the result of removing from $\mathcal{G}$ all edges $(u, r_\star, u)$, and all edges $(u, r_\star, u')$ such that (u, r, u') is also in $\mathcal{G}$.*

The goal of the $\mathcal{G}_\star$ construction is to make any graph $\star$-closed. Note that $\mathcal{G}_\star$ contains 'loops' $(u, r_\star, u)$ for all nodes u and roles r, but its redundant star removal does not.

Lemma 4. *$\mathcal{G}_\star$ is $\star$-closed for every graph $\mathcal{G}$. If a given $\mathcal{G}$ is $\star$-closed, so is $\mathsf{rsr}(\mathcal{G})$.*

Given a graph $\mathcal{G} = (V, E, \ell)$ and nodes $\vec{v} = (v_1, \dots v_n)$ in V, we let $\Pi(\mathcal{G}, \vec{v})$ be the product of the pointed graphs $(\mathcal{G}, v_i)$, defined as usual. We may call $\vec{v}$ the distinguished element of $\Pi(\mathcal{G}, \vec{v})$.

We define our *star product*, which preserves r^*-paths. In a nutshell, we make the input $\star$-closed, take the regular product, and remove unnecessary stars; we restrict it to the given signature as usual.

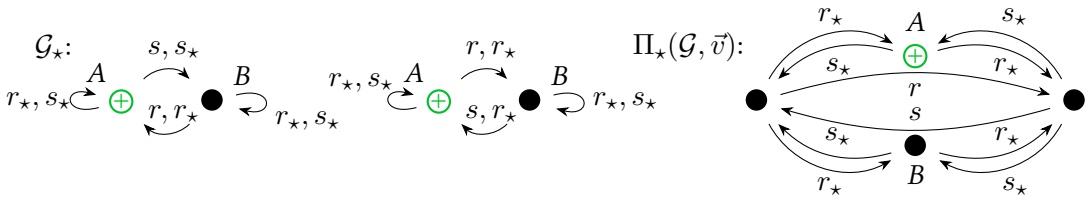
Definition 8 (star product)**.** *Given $\mathcal{G}$, nodes $\vec{v} = (v_1, \dots v_n)$ in $\mathcal{G}$, and a signature Σ, we denote by $\Pi_\star(\mathcal{G}, \vec{v}, \Sigma)$ the pointed graph $\mathsf{rsr}(\Pi(\mathcal{G}_\star, \vec{v}))|_{\Sigma^\star_\mathsf{R}}$. If $\Sigma = \Sigma(\mathcal{G})$, we may omit it and write simply $\Pi_\star(\mathcal{G}, \vec{v})$.*

The star product is a pointed starred graph with distinguished node $\vec{v}$. Importantly, it is $\star$-closed.

Lemma 5. *$\Pi_\star(\mathcal{G}, \vec{v}, \Sigma)$ is $\star$-closed for every $\mathcal{G}$, $\vec{v}$ and Σ.*

Proof sketch. Let $\Pi_\star(\mathcal{G}, \vec{v}, \Sigma)$ be the star product graph of some graph $\mathcal{G}$ with distinguished node $\vec{v}$ and signature Σ. For any node v in $\Pi_\star(\mathcal{G}, \vec{v}, \Sigma)$ (which could be $\vec{v}$ or any other node), by definition v appears in every pointed graph $(\mathcal{G}, v_i)$ used to construct $\Pi_\star(\mathcal{G}, \vec{v}, \Sigma)$. Furthermore, if there exists a node v' in $\Pi_\star(\mathcal{G}, \vec{v}, \Sigma)$ s.t. $(v, v') \in (r \cup r_\star)^*$ and $(v, v') \notin r$, then there also exists such a path between v and v' in each pointed graph $(\mathcal{G}, v_i)$. There then also exists an $r_\star$-edge from v to v' in each $(\mathcal{G}_\star, v_i)$, where $\mathcal{G}_\star$ is the star-closed version of $\mathcal{G}$, by definition. Since this edge exists in each such pointed graph and $r_\star \in \Sigma^\star_\mathsf{R}$, it is preserved when we take their product. Note that removing "redundant" $r_\star$-edges from this product still allows the graph to be $\star$-closed and results in the star product $\Pi_\star(\mathcal{G}, \vec{v}, \Sigma)$ as defined. Thus $\Pi_\star(\mathcal{G}, \vec{v}, \Sigma)$ is still $\star$-closed. □

Example 5. *Consider Example 4 again. We first extend $\mathcal{G}$ to $\mathcal{G}_\star$. Then, we obtain $\Pi_\star(\mathcal{G}, \vec{v}) = \mathsf{rsr}(\Pi(\mathcal{G}_\star, \vec{v}))$, with the node $\vec{v}$ marked green:*



Simulations. Intuitively, the $\mathcal{EL}^*$ concepts that are satisfied in the star product are exactly those that are satisfied in all the pointed graphs. To formalise and prove this, we define a notion of simulation that captures the semantics of $\mathcal{EL}^*$ over possibly starred graphs.

Definition 9. *An $\mathcal{EL}^*$ Σ-simulation ζ between two (possibly starred) graphs $\mathcal{I} = (V_\mathcal{I}, E_\mathcal{I}, \ell_\mathcal{I})$ and $\mathcal{G} = (V_\mathcal{G}, E_\mathcal{G}, \ell_\mathcal{G})$ and given a signature Σ, is a relation $\zeta \subseteq V_\mathcal{I} \times V_\mathcal{G}$, which satisfies the properties **AtomR**, **Forth**, and **Forth***, defined below.*

AtomR *If $(d, e) \in \zeta$ and $d \in A^\mathcal{I}$, then $e \in A^\mathcal{G}$.*

Forth *For every $r \in \Sigma^\star_\mathsf{R}$, if $(d, e) \in \zeta$ and $(d, d') \in r^\mathcal{I}$, then there is $(e, e') \in r^\mathcal{G}$ with $(d', e') \in \zeta$.*

Forth* *For every $r \in \Sigma^\star_\mathsf{R}$, if $(d, e) \in \zeta$ and $(d, d') \in ((r \cup r_\star)^*)^\mathcal{I}$, then there is $(e, e') \in ((r \cup r_\star)^*)^\mathcal{G}$ with $(d', e') \in \zeta$.*

Given two nodes $d \in V_\mathcal{I}$ and $e \in V_\mathcal{G}$, we write $(\mathcal{I}, d) \preceq_\Sigma (\mathcal{G}, e)$ if there is a Σ-simulation ζ where $(d, e) \in \zeta$. If Σ is $\Sigma(\mathcal{I})$, then we may omit it and write $(\mathcal{I}, d) \preceq (\mathcal{G}, e)$.

$\mathcal{EL}^*$ Σ-simulations capture the semantics of $\mathcal{EL}^*$ concepts in the usual way.

Lemma 6. *The following hold for all pointed graphs and signatures Σ:*

1. *If $(\mathcal{I}, d) \preceq_\Sigma (\mathcal{G}, e)$, then for every $\mathcal{EL}^*$ Σ-concept C, if $d \in C^\mathcal{I}$ then $e \in C^\mathcal{G}$.*
2. *For every $\mathcal{EL}^*$ concept C, if $d \in C^\mathcal{I}$, then $\mathcal{T}_C \preceq (\mathcal{I}, d)$.*
3. *For all $\mathcal{EL}^*$ concepts C and D, we have $C \sqsubseteq D$, iff $\mathcal{T}_D \preceq \mathcal{T}_C$.*

We characterise the correctness of our product construction in terms of $\mathcal{EL}^*$ Σ-simulations.

Lemma 7. *Given a graph $\mathcal{G}$, a signature Σ and nodes $\vec{v} = v_1, ...v_n$ in $\mathcal{G}$ for some arbitrary n, for every $(\mathcal{I}, d)$ we have that $(\mathcal{I}, d) \preceq_\Sigma (\mathcal{G}, v_1), \ldots, (\mathcal{I}, d) \preceq_\Sigma (\mathcal{G}, v_n)$ if and only if $(\mathcal{I}, d) \preceq_\Sigma \Pi_\star(\mathcal{G}, \vec{v}, \Sigma)$.*

If the star product of all the positive examples, which represents their commonalities, is a fitting, then it is the MSF. However, to be a fitting, we need to make sure that, on the one hand, it does not simulate any negative example, and on the other, that it can be represented as a (finite) concept. We will now prove that the latter holds exactly when its unravelling is finite.

Σ-Unravellings. Unravelling is a well-known technique for transforming an arbitrary graph into a tree-shaped one. Given a (possibly starred) graph $\mathcal{G} = (V, E, \ell)$ and a signature Σ, *a $\Sigma^\star_\mathsf{R}$-path p (of length n)* is a sequence $p = v_0 r_0 v_1 \ldots v_{n-1} r_{n-1} v_n$, with $v_i \in V$, $r_i \in \Sigma^\star_\mathsf{R}$ and for every r_i with $1 \leq i < n$ we have that $(v_i, v_{i+1}) \in (r_i)^\mathcal{G}$. We say that p *starts with* v_0 and *ends with* v_n, and we may write $tail(p) = v_n$. We write pRb to mean a $\Sigma^\star_\mathsf{R}$-path that extends p by following an edge labelled with R to a node b. We may omit $\Sigma^\star_\mathsf{R}$ and speak of just paths when $\Sigma^\star_\mathsf{R}$ is clear from context.

Let $v \in V$. The *Σ-unravelling of v in $\mathcal{G}$* is the graph $\mathcal{U}_{v,\Sigma} = (V_\mathcal{U}, E_\mathcal{U}, \ell_\mathcal{U})$ such that:

- $V_\mathcal{U}$ contains all $\Sigma^\star_\mathsf{R}$-paths that start with v.
- $A \in \ell_\mathcal{U}(p)$ for all p s.t. $A \in \ell(tail(p)) \cap \Sigma$
- $(p, R, pRb) \in E_\mathcal{U}$ for every path $pRb \in V_\mathcal{U}$.

If we add the requirement that $V_\mathcal{U}$ contains only $\Sigma^\star_\mathsf{R}$-paths of length at most m for some $m \geq 0$, then we obtain an *m-bounded Σ-unravelling*, or just m-Σ-unravelling, denoted $\mathcal{U}^m_{v,\Sigma}$. Note that if m is at least the length of the longest path, then $\mathcal{U}^m_{v,\Sigma} = \mathcal{U}_{v,\Sigma}$. If this is not the case, we say that $\mathcal{U}^m_{v,\Sigma}$ is *strict*. If Σ is $\Sigma(\mathcal{G})$, we may omit it and simply write $\mathcal{U}_v$.

Note that a starred graph unravels into a starred tree-shaped graph. As for description trees, we might simply write $\mathcal{U}_v$ to denote the pointed graph $(\mathcal{U}_v, v)$ of an unravelling $\mathcal{U}_v$ and its root v.

Example 6. *Consider again Example 4. The unravelling $\mathcal{U}_{\vec{v}}$ of $\Pi_\star(\mathcal{G}, \vec{v})$ is an infinite tree-shaped graph where each node has either one outgoing $r_\star$-edge and one $s_\star$-edge, or two outgoing $r_\star$-edges and one r-edge, or two outgoing $s_\star$-edges and one s-edge.*

The following lemma connects our notions of simulation and unravelling.

Lemma 8. *Given two pointed graphs $(\mathcal{I}, d)$ and $(\mathcal{G}, e)$, we have 1. $(\mathcal{I}, d) \preceq (\mathcal{G}, e)$ iff $(\mathcal{U}_d, d) \preceq (\mathcal{G}, e)$, and 2. If $\mathcal{I}$ is a tree, then $(\mathcal{I}, d) \preceq_\Sigma (\mathcal{G}, e)$ iff $(\mathcal{I}, d) \preceq_\Sigma (\mathcal{U}_e, e)$ for every Σ.*

Unravellings of $\star$-closed graphs are in general not $\star$-closed: they are tree-shaped, and do not have '$r_\star$-shortcuts' to witness longer r^*-paths. However, if the unravelled graph is $\star$-closed, and if a node reaches another along an r^*-path in the unravelling, it will also have a similar 'copy' as an immediate $r_\star$-child. This is useful, since it allows us to find simulations by looking at bounded unravellings.

Lemma 9. *For all Σ and all $(\mathcal{I}, d)$ and $(\mathcal{G}, e)$ such that $\mathcal{G}$ is $\star$-closed,*

1. *For every $\mathcal{EL}^*$ concept C with $\Sigma(C) \subseteq \Sigma$ s.t. $\mathcal{T}_C \preceq \mathcal{U}_e$, we have $\mathcal{T}_C \preceq \mathcal{U}^m_e$, where $m = depth(C)$.*
2. *If $\mathcal{I}$ and $\mathcal{G}$ are finite, then $\mathcal{U}_d \preceq_\Sigma (\mathcal{G}, e)$ iff $\mathcal{U}^m_d \preceq_\Sigma (\mathcal{G}, e)$ for all m-unravellings $\mathcal{U}^m_d$ of $\mathcal{I}$ at d.*

Proof sketch. Item 2 follows from the proof of Lemma 5.5 in [11], but adapted from the $\mathcal{ELI}$ setting to $\mathcal{EL}^*$ Σ-simulations and the semantics of starred graphs. For the $\mathcal{EL}^*$ setting, if we additionally require that $\mathcal{G}$ is $\star$-closed, then the proof immediately extends as expected.

For Item 1, the proof uses pre-matches (which are Σ-simulations that are additionally functional). The key idea is that if there exists a pre-match ρ from a node u_p in a description tree into $\mathcal{U}_v$, such that if u_p has an immediate $r_\star$-child u_c in the description tree and its pre-match $\rho(u_p)$ is such that it is connected to $\rho(u_v)$ via more than one r or $r_\star$-edges in $\mathcal{U}_v$, then we can use the fact that $\mathcal{G}$ is $\star$-closed to show that $\rho(u_p)$ also has an immediate $r_\star$-child that has the same label as $\rho(u_v)$ and that is the root of a subtree in $\mathcal{U}_v$ isomorphic to the one rooted at $\rho(u_c)$. From this observation, it is not hard to see that we find a Σ-simulation that does not use paths longer than m. □

Algorithm for computing MSF in $\mathcal{EL}^*$. In the following, $\vec{v}_+$ denotes an (arbitrary but fixed) enumeration of the positive nodes $\mathscr{P}$ of a fitting instance $\mathcal{F} = (\mathcal{G}, \mathscr{P}, \mathscr{N}, \Sigma)$. Our algorithms for deciding existence of an MSF and computing it build on the following proposition, similar to Prop. 5.14 in [11].

Proposition 2. *Let $\mathcal{F} = (\mathcal{G}, \mathscr{P}, \mathscr{N}, \Sigma)$ and let C be an $\mathcal{EL}^*$ concept. The following are equivalent: 1) C is a most specific fitting for $\mathcal{F}$, and 2) C is a fitting for $\mathcal{F}$ and $\Pi_\star(\mathcal{G}, \vec{v}_+, \Sigma) \preceq \mathcal{T}_C$.*

Proof. (1) $\Rightarrow$ (2). Assume that C is an $\mathcal{EL}^*$ MSF. Then C fits all $v_i \in \mathscr{P}$, so by Lemma 6 Item 2, $\mathcal{T}_C \preceq (\mathcal{G}, v_i)$ for all $v_i \in \mathscr{P}$. By Lemma 7, then $\mathcal{T}_C \preceq \Pi_\star(\mathcal{G}, \vec{v}_+, \Sigma)$ and by Lemma 8 Item 2, $\mathcal{T}_C \preceq \mathcal{U}_{\vec{v}_+}$.

Let $m = depth(C)$ and let $\mathcal{U}^m_{\vec{v}_+}$ be the m-bounded unravelling of $\Pi_\star(\mathcal{G}, \vec{v}_+, \Sigma)$. Then by Lemma 9 Item 1, $\mathcal{T}_C \preceq \mathcal{U}^m_{\vec{v}_+}$ Suppose that $\mathcal{U}_{\vec{v}_+} \preceq (\mathcal{G}, n_i)$ for some $n_i \in \mathscr{N}$. Then $\Pi_\star(\mathcal{G}, \vec{v}_+, \Sigma) \preceq (\mathcal{G}, n_i)$ by Lemma 8 Item 1 and, since $\mathcal{T}_C \preceq \Pi_\star(\mathcal{G}, \vec{v}_+, \Sigma)$, we have $\mathcal{T}_C \preceq (\mathcal{G}, n_i)$. This is a contradiction since C is a fitting for $\mathcal{F}$. So we must have $\mathcal{U}_{\vec{v}_+} \not\preceq (\mathcal{G}, n_i)$ for any $n_i \in \mathscr{N}$. Then by Lemma 9 Item 2, $\mathcal{U}^m_{\vec{v}_+} \not\preceq (\mathcal{G}, n_i)$ for any $n_i \in \mathscr{N}$. Thus the corresponding concept C_m of $\mathcal{U}^m_{\vec{v}_+}$ is a fitting for $\mathcal{F}$. This same argument can then also be extended for any m'-unravelling s.t. $m' \geq m$. Hence the corresponding concept $C_{m'}$ of $\mathcal{U}^{m'}_{\vec{v}_+}$ is also a fitting for $\mathcal{F}$. Since C is an MSF, for any such $C_{m'}$ we have $C \sqsubseteq C_{m'}$. By Lemma 6 Item 3, then $\mathcal{U}^{m'}_{\vec{v}_+} \preceq \mathcal{T}_C$ for any bounded unravelling s.t. $m' \geq m$. In particular, this then also holds for the full unravelling $\mathcal{U}_{\vec{v}_+}$. So we have $\mathcal{U}_{\vec{v}_+} \preceq \mathcal{T}_C$. It then follows from Lemma 8 Item 1 that $\Pi_\star(\mathcal{G}, \vec{v}_+, \Sigma) \preceq \mathcal{T}_C$.
(2) $\Rightarrow$ (1). Assume C is a fitting for $\mathcal{F}$ and $\Pi_\star(\mathcal{G}, \vec{v}_+, \Sigma) \preceq \mathcal{T}_C$. Let D be any $\mathcal{EL}^*$ fitting for $\mathcal{F}$. Then by Lemma 6 Item 2, $\mathcal{T}_D \preceq (\mathcal{G}, v_i)$ for all $v_i \in \mathscr{P}$. Therefore by Lemma 7, $\mathcal{T}_D \preceq \Pi_\star(\mathcal{G}, \vec{v}_+, \Sigma)$. Then we have $\mathcal{T}_D \preceq \mathcal{T}_C$, and thus by Lemma 6 Item 3, $C \sqsubseteq D$. Therefore C is an MSF for $\mathcal{F}$. □

We now have an algorithm for deciding MSF existence.

Theorem 3. *Given $\mathcal{F} = (\mathcal{G}, \mathscr{P}, \mathscr{N}, \Sigma)$, an $\mathcal{EL}^*$ most specific fitting exists iff the unravelling of $\Pi_\star(\mathcal{G}, \vec{v}_+, \Sigma)$ is finite and its corresponding concept D is an $\mathcal{EL}^*$ fitting for $\mathcal{F}$.*

Proof. First, assume that C is an $\mathcal{EL}^*$ MSF for $\mathcal{F}$. By Proposition 2, $\Pi_\star(\mathcal{G}, \vec{v}_+, \Sigma) \preceq \mathcal{T}_C$. Let $m = depth(C)$ and let $\mathcal{U}^m_{\vec{v}_+}$ be the m-unravelling of $\Pi_\star(\mathcal{G}, \vec{v}_+, \Sigma)$. Then by the fact that $\Pi_\star(\mathcal{G}, \vec{v}_+, \Sigma)$ is $\star$-closed and by Lemma 9 Item 1, $\mathcal{T}_C \preceq \mathcal{U}^m_{\vec{v}_+}$. Therefore by Proposition 2, $\Pi_\star(\mathcal{G}, \vec{v}_+, \Sigma) \preceq \mathcal{U}^m_{\vec{v}_+}$, and the corresponding concept of $\mathcal{U}^m_{\vec{v}_+}$ is a fitting for $\mathcal{F}$. By Proposition 2, the corresponding concept of $\mathcal{U}^m_{\vec{v}_+}$ is the MSF C. We apply the following claim, proven in the appendix: (††) If $\mathcal{U}^n_{\vec{v}_+}$ is a strict n-bounded unravelling for some $n \geq 0$, then $\mathcal{U}_{\vec{v}_+} \not\preceq \mathcal{U}^n_{\vec{v}_+}$. In a nutshell, for any strict bounded unravelling, we can always unravel the graph further and doing so will result in an unravelling that is more specific than the previous. Thus, if we assume that $\mathcal{U}^m_{\vec{v}_+}$ is strict, then there exists $m' > m$ s.t. $\mathcal{U}^m_{\vec{v}_+} \preceq \mathcal{U}^{m'}_{\vec{v}_+}$. This implies, by Lemma 6 Item 3, that there exists a concept C' (whose corresponding $\mathcal{EL}^*$ tree is equivalent to $\mathcal{U}^{m'}_{\vec{v}_+}$) s.t. $C' \sqsubseteq C$. Furthermore, since $m' > m$, we know that the longest path in $\mathcal{U}^m_{\vec{v}_+}$ is strictly shorter than the longest path $\mathcal{U}^{m'}_{\vec{v}_+}$. Therefore, it cannot be the case that $C \sqsubseteq C'$. This contradicts that C is the MSF. Hence it must be the case that $\mathcal{U}^m_{\vec{v}_+}$ is not strict, i.e. we have that $\mathcal{U}^m_{\vec{v}_+} = \mathcal{U}_{\vec{v}_+}$. Therefore $\mathcal{U}_{\vec{v}_+}$ is finite and its corresponding concept is the MSF C.

For the other direction, assume that $\mathcal{U}_{\vec{v}_+}$ is finite and that its corresponding concept D is an $\mathcal{EL}^*$ fitting for $\mathcal{F}$. Then, since $\mathcal{U}_{\vec{v}_+} \preceq \mathcal{T}_D$ (in fact the two are equivalent, $\mathcal{U}_{\vec{v}_+}$ is the description tree of D) and by Lemma 8 Item 1, we have $\Pi_\star(\mathcal{G}, \vec{v}_+, \Sigma) \preceq \mathcal{T}_D$. By Proposition 2, D is an $\mathcal{EL}^*$ MSF for $\mathcal{F}$. □

We can now prove that the existence of an $\mathcal{EL}^*$ MSF implies the existence of an $\mathcal{EL}$ MSF.

Proposition 3. *If an $\mathcal{EL}^*$ MSF exists for some $\mathcal{F}$, then an $\mathcal{EL}$ MSF exists for $\mathcal{F}$.*

Proof. Let C be an MSF for some fitting instance $\mathcal{F}$ with the Σ-unravelling $\mathcal{U}_{\vec{v}_+}$.Then the statement follows from Theorem 3, together with the construction of $\mathcal{U}_{\vec{v}_+}$. Let $\mathcal{U}'_{\vec{v}_+}$ be $\mathcal{U}_{\vec{v}_+}$ with all paths containing $r_\star$-labelled edges removed. Then $\mathcal{U}'_{\vec{v}_+}$ is non-starred and contained in $\mathcal{U}_{\vec{v}_+}$. By Theorem 3, $\mathcal{U}_{\vec{v}_+}$ is finite and its corresponding concept D is a fitting for $\mathcal{F}$. Then $\mathcal{U}'_{\vec{v}_+}$ is also finite and its corresponding concept D' is such that $D \sqsubseteq D'$. Therefore D' is also a fitting for $\mathcal{F}$. The proposition then follows from Theorem 3, restricted to non-starred unravellings and $\mathcal{EL}$ concepts. □

Algorithm 1: Computation of an fcu-product

Input: $\mathcal{F} = (\mathcal{G}, \mathscr{P}, \mathscr{N}, \Sigma)$: a fitting instance
Output: An fcu-product for $\mathcal{F}$ if it exists, or $\emptyset$ otherwise.

function `computeProd(`$\mathcal{F}$`)`:
  $\vec{v}_+ \coloneqq (v_0, \ldots, v_\ell)$ with $v_i \in \mathscr{P}$ and $1 \leq i \leq \ell$
  $\Pi \coloneqq \Pi_\star(\mathcal{G}, \vec{v}_+, \Sigma)$
  if `cycleDetect(`Π`)` = *true* **then**
    return $\emptyset$
  foreach $n \in \mathscr{N}$ **do**
    if $\Pi \preceq (\mathcal{G}, n)$ **then**
      return $\emptyset$
  return Π

Algorithm 2: Computation of an $\mathcal{EL}^*$ MSF

Input: $\mathcal{F} = (\mathcal{G}, \mathscr{P}, \mathscr{N}, \Sigma)$: a fitting instance
Output: An $\mathcal{EL}^*$ MSF for $\mathcal{F}$ if it exists, or $\emptyset$ otherwise.

function `computeMSF(`$\mathcal{F}$`)`:
  $\Pi \coloneqq$ `computeProd(`$\mathcal{F}$`)`
  if $\Pi = \emptyset$ **then**
    return $\emptyset$
  $\mathcal{U} \coloneqq \Sigma$-unravelling of Π
  return corresponding concept of $\mathcal{U}$

As illustrated in Example 4, the converse of Proposition 3 fails: the existence of an $\mathcal{EL}$ MSF does not imply the existence of an $\mathcal{EL}^*$ MSF. We showed in Example 6 that the Σ-unravelling $\mathcal{U}_{\vec{v}_+}$ for $\mathcal{F}$ is infinite. By Theorem 3, this implies that an $\mathcal{EL}^*$ MSF for $\mathcal{F}$ does not exist, as anticipated.

We call $\Pi_\star(\mathcal{G}, \vec{v}_+, \Sigma)$ a *fitting product* if $\Pi_\star(\mathcal{G}, \vec{v}_+, \Sigma) \not\preceq (\mathcal{G}, n)$ for all $n \in \mathscr{N}$. If additionally *no cycle is reachable from* $\vec{v}^+$, we call it *concept unravellable*, and we call it a *fcu-product* if it is both.

We are now ready to present a computation algorithm of $\mathcal{EL}^*$ MSFs given a fitting instance $\mathcal{F}$. Algorithm 1 returns the fcu-product $\Pi \coloneqq \Pi_\star(\mathcal{G}, \vec{v}_+, \Sigma)$, if it exists, or returns $\emptyset$ otherwise. Note that the function `cycleDetect` in Algorithm 1 takes as input a pointed graph $(\mathcal{G}, v)$ and returns *true* iff $\mathcal{G}$ has a cycle reachable from v. If no such cycle is found, we know that the unravelling of Π is finite. Algorithm 1 then checks whether Π simulates into the pointed graphs of any of the negative nodes. If so, it returns $\emptyset$, otherwise it returns Π as the fcu-product. If the fcu-product exists, then Algorithm 2 constructs the corresponding $\mathcal{EL}^*$ concept C of the Σ-unravelling $\mathcal{U}$ of Π and returns it.

Theorem 4. *Given a fitting instance $\mathcal{F} = (\mathcal{G}, \mathscr{P}, \mathscr{N}, \Sigma)$, it holds that:*

1. *Algorithm 1 returns an fcu-product iff an $\mathcal{EL}^*$ MSF for $\mathcal{F}$ exists, and it runs in ExpTime.*
2. *Algorithm 2 returns an $\mathcal{EL}^*$ concept C iff C is an $\mathcal{EL}^*$ MSF for $\mathcal{F}$.*
3. *If $|\mathscr{P}| \leq k$ for a constant k, then one can decide the existence of an $\mathcal{EL}^*$ MSF in polynomial time.*

Proof. (1) Let V be the nodes of $\mathcal{G}$. It is well-known that the standard product $\Pi(\mathcal{G}, \vec{v})$ can be constructed in time that is bounded by a polynomial in $|V|$ and an exponential in $|\vec{v}|$, but the construction only needs polynomial time if $|\vec{v}|$ is a constant. Since our star product only adds pre- and post-processing steps that are polynomial in $|V|$, these bounds apply as well. `cycleDetect` runs in time polynomial to $|\Pi_\star(\mathcal{G}, \vec{v}_+, \Sigma)|$, which in general may be of exponential size. Then, Algorithm 1 checks whether $\Pi_\star(\mathcal{G}, \vec{v}_+, \Sigma)$ simulates into the pointed graph of any of the negative examples in $\mathscr{N}$, this only needs polynomial time. Algorithm 1 will then conclude by outputting the fcu-product. By Theorem 3, the fcu-product exists iff an MSF exists; since the MSF exists iff there is a finite unravelling of the fitting product, and since the fcu-product contains no cycles it will have a finite unravelling. Thus Algorithm 1 decides the existence of an MSF in exponential time.

(2) Follows directly from Theorem 3. To show (3), note that `cycleDetect` runs in time polynomial to $|\Pi_\star(\mathcal{G}, \vec{v}_+, \Sigma)|$ which is clearly polynomial if $|\mathscr{P}| \leq k$. Thus in this case Algorithm 1 checks existence of the fcu-product in polynomial time. If this exists, then an MSF exists. □

The unravelling of the fcu-product $\Pi_\star(\mathcal{G}, \vec{v}_+, \Sigma)$ can be exponentially larger than $\Pi_\star(\mathcal{G}, \vec{v}_+, \Sigma)$, hence the MSF concept may be of double exponential size, and exponential even when $\Pi_\star(\mathcal{G}, \vec{v}_+, \Sigma)$ is polynomial. However, $\Pi_\star(\mathcal{G}, \vec{v}_+, \Sigma)$ itself is a succinct graph representation of the MSF when it exists.

6. Conclusion

We have conducted an initial principled study into the problem of learning arbitrary and most specific fitting concepts in the Description Logic $\mathcal{EL}^*$ for a set of positive and negative examples on interpretations, represented as graphs. We have shown that for this DL, the fitting existence problem is ExpTime-complete. Furthermore, for $\mathcal{EL}^*$ MSFs, we developed a decision procedure for deciding existence that runs in exponential time, but that needs only polynomial time if the number of positive examples is bounded by a constant. We also presented an algorithm to compute an MSF.

For future research, we would like to expand the fitting target language further towards SHACL. Although full disjunction and node tests (which amount to nominals in DLs) trivialise the fitting problem, there are many extensions that have not been considered much in the DL community for concept learning, but that are highly relevant in the SHACL setting. A natural step is to adopt inverse roles Fitting problems with inverses have already been studied in the context of $\mathcal{ELI}$, for instance in [11], and have been shown to change the behaviour of the algorithms significantly, which would likely extend to our setting. Other features like counting over paths, path equality, and closedness seem exciting but also challenging and would require new techniques to be developed. In this paper, we have limited ourselves to studying arbitrary and most specific fittings for $\mathcal{EL}^*$. Other problems related to fittings, such as computing most general, succinct, and bounded fittings for $\mathcal{EL}^*$ remain unexplored, but it would certainly make sense to investigate these in future work.

Acknowledgments

This research was funded in whole or in part by the Austrian Science Fund (FWF) 10.55776/COE12 and 10.55776/PIN8884924.

Declaration on Generative AI

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

References

[1] F. Baader, R. Küsters, R. Molitor, Computing least common subsumers in description logics with existential restrictions, in: T. Dean (Ed.), Proceedings of the 16th International Joint Conference on Artificial Intelligence, IJCAI, Morgan Kaufmann, 1999, pp. 96–103.

[2] S. Brandt, A. Turhan, R. Küsters, Extensions of non-standard inferences to descriptions logics with transitive roles, in: M. Y. Vardi, A. Voronkov (Eds.), Proceedings of 10th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning, LPAR, LNCS, Springer, 2003, pp. 122–136.

[3] F. Baader, B. Sertkaya, A. Turhan, Computing the least common subsumer w.r.t. a background terminology, J. Appl. Log. 5 (2007) 392–420.

[4] B. Zarrieß, A. Turhan, Most specific generalizations w.r.t. general $\mathcal{EL}$-tboxes, in: F. Rossi (Ed.), Proc. IJCAI 2013, IJCAI/AAAI, 2013, pp. 1191–1197. URL: http://www.aaai.org/ocs/index.php/IJCAI/IJCAI13/paper/view/6709.

[5] F. Distel, Learning description logic knowledge bases from data using methods from formal concept analysis, Ph.D. thesis, Dresden University of Technology, 2011.

[6] B. ten Cate, R. Koudijs, A. Ozaki, On the power and limitations of examples for description logic concepts, in: Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI, ijcai.org, 2024, pp. 3567–3575.

[7] M. Funk, J. C. Jung, C. Lutz, H. Pulcini, F. Wolter, Learning description logic concepts: When can positive and negative examples be separated?, in: S. Kraus (Ed.), Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, ijcai.org, 2019, pp. 1682–1688.

[8] B. ten Cate, M. Funk, J. C. Jung, C. Lutz, SAT-based PAC learning of description logic concepts, in: Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI '23, 2023. URL: https://doi.org/10.24963/ijcai.2023/373. doi:10.24963/ijcai.2023/373.

[9] B. ten Cate, M. Funk, J. C. Jung, C. Lutz, On the non-efficient PAC learnability of conjunctive queries, Information Processing Letters 183 (2024) 106431. URL: https://www.sciencedirect.com/science/article/pii/S0020019023000741. doi:https://doi.org/10.1016/j.ipl.2023.106431.

[10] B. ten Cate, M. Funk, J. Christoph Jung, C. Lutz, Fitting algorithms for conjunctive queries, SIGMOD Rec. 52 (2024) 6–18. URL: https://doi.org/10.1145/3641832.3641834. doi:10.1145/3641832.3641834.

[11] B. ten Cate, V. Dalmau, M. Funk, C. Lutz, Extremal fitting problems for conjunctive queries, CoRR abs/2206.05080 (2022). URL: https://doi.org/10.48550/arXiv.2206.05080. doi:10.48550/ARXIV.2206.05080. arXiv:2206.05080.

[12] S. Tirtarasa, A. Turhan, Computing generalizations of temporal $\mathcal{EL}$ concepts with next and global, in: J. Hong, M. Bures, J. W. Park, T. Cerný (Eds.), SAC '22: The 37th ACM/SIGAPP Symposium on Applied Computing, ACM, 2022, pp. 903–910.

[13] J. C. Jung, V. Ryzhikov, F. Wolter, M. Zakharyaschev, Extremal separation problems for temporal instance queries, in: Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI 2024, Jeju, South Korea, August 3-9, 2024, ijcai.org, 2024, pp. 3448–3456.

[14] H. Knublauch, D. Kontokostas, Shapes constraint language (SHACL), W3C Recommendation, 2017. https://www.w3.org/TR/shacl/.

[15] I. Boneva, J. Dusart, D. Fernández-Álvarez, J. E. L. Gayo, Shape designer for ShEx and SHACL constraints, in: M. C. Suárez-Figueroa, G. Cheng, A. L. Gentile, C. Guéret, C. M. Keet, A. Bernstein (Eds.), Proceedings of the ISWC 2019 Satellite Tracks co-located with 18th (ISWC 2019), volume 2456 of *CEUR Workshop Proceedings*, 2019, pp. 269–272.

[16] D. Fernández-Álvarez, J. E. L. Gayo, D. Gayo-Avello, Automatic extraction of shapes using sheXer, Knowl. Based Syst. 238 (2022) 107975. URL: https://doi.org/10.1016/j.knosys.2021.107975. doi:10.1016/J.KNOSYS.2021.107975.

[17] P. G. Omran, K. Taylor, S. J. R. Méndez, A. Haller, Learning SHACL shapes from knowledge graphs, Semantic Web 14 (2023) 101–121. URL: https://doi.org/10.3233/SW-223063. doi:10.3233/SW-223063.

[18] B. Bogaerts, M. Jakubowski, J. Van den Bussche, SHACL: A description logic in disguise, in: Proc. of LPNMR 2022, volume 13416 of *Lecture Notes in Computer Science*, Springer, 2022, pp. 75–88. URL: https://doi.org/10.1007/978-3-031-15707-3_7. doi:10.1007/978-3-031-15707-3_7.

[19] M. Ortiz, A short introduction to shacl for logicians, in: H. H. Hansen, A. Scedrov, R. J. de Queiroz (Eds.), Logic, Language, Information, and Computation, Springer Nature Switzerland, Cham, 2023, pp. 19–32.

[20] F. Baader, R. Küsters, R. Molitor, Computing least common subsumers in description logics with existential restrictions, in: Proceedings of the 16th International Joint Conference on Artifical Intelligence - Volume 1, Morgan Kaufmann Publishers Inc., 1999, p. 96–101.

[21] D. Calvanese, G. D. Giacomo, M. Lenzerini, 2ATAs make DLs easy, in: I. Horrocks, S. Tessaris (Eds.), Proceedings of the 2002 International Workshop on Description Logics, volume 53 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2002.

[22] M. Y. Vardi, Reasoning about the past with two-way automata, in: K. G. Larsen, S. Skyum, G. Winskel (Eds.), Automata, Languages and Programming, Springer Berlin Heidelberg, Berlin, Heidelberg, 1998, pp. 628–641.

[23] D. E. Muller, P. E. Schupp, Alternating automata on infinite trees, Theor. Comput. Sci. 54 (1987) 267–276. URL: https://doi.org/10.1016/0304-3975(87)90133-2. doi:10.1016/0304-3975(87)90133-2.

A. Proofs for Section 4 (Complexity of Fitting Existence)

Definition 10 (k-ATA Run). *A* run *of a k-ATA $\mathfrak{A}$ over a node-labelled tree $\mathfrak{T}_C = (V, \ell)$ with degree k is a $(V \times Q)$-labelled tree $\mathfrak{T}_\mathsf{r} = (V_\mathsf{r}, \ell_\mathsf{r})$ satisfying:*

1. *$\varepsilon \in V_\mathsf{r}$ and $\ell_\mathsf{r}(\varepsilon) = (\varepsilon, q_0)$.*
2. *Let $y \in V_\mathsf{r}$ with $\ell_\mathsf{r}(y) = (x, q)$ and $\delta(q, \ell(x)) = C$. Then there is a (possibly empty) set $S = \{(c_1, q_1), \dots, (c_n, q_n)\} \subseteq [k] \times Q$ such that:*
   - *S satisfies C, and*
   - *for all $1 \leq i \leq n$, we have that $y \cdot i \in V_\mathsf{r}$, $x \cdot c_i \in V$ and $\ell_\mathsf{r}(y \cdot i) = (x \cdot c_i, q_i)$.*

Given an infinite path $p \subseteq V_\mathsf{r}$, let $inf(p) \subseteq Q$ be the set of states that appear infinitely often in p. A run $\mathfrak{T}_\mathsf{r}$ of a ATA $\mathfrak{A} = \langle \Sigma, Q, \delta, q_0, p \rangle$ is accepting *if for every infinite path $p \subseteq V_\mathsf{r}$, the state with the highest priority in $inf(p)$ is even. Note that a finite run is always accepting.*

A partial run *of a k-ATA $\mathfrak{A}$ is a run that does not satisfy Condition 1 above. If a partial run has as its root a state (w, q), we call it a* partial run rooted at (w, q)*, or, for short, a* partial (w, q)-run.

Lemma 2. *Given a pointed graph $(\mathcal{G}, v_0)$, the k-ATA $\mathfrak{A}^{\mathcal{G}, v_0}$ accepts a proper node-labelled $\mathcal{EL}^*$ k-tree $\mathfrak{T}$ iff it has a pre-match into $(\mathcal{G}, v_0)$.*

Proof. Let $\mathfrak{T}$ be a proper node-labelled $\mathcal{EL}^*$ k-tree with root ε, and let $(\mathcal{G}, v_0)$ be a pointed graph. $\mathfrak{T}$ has an *r-reachable* pre-match into $(\mathcal{G}, v)$ if there exists some v' in $\mathcal{G}$ such that there exists a pre-match ρ from w in $\mathfrak{T}$ into $(\mathcal{G}, v')$ s.t. $\rho(w) = v'$ and v' can be reached from v via a (possibly empty) path of r-edges.

We show the following:

1. Let $w \in \mathfrak{T}$ and let $w \neq \varepsilon$, and let $q_- \in Q$. There exists a partial (w, q_-)-run iff $w.R = -$. Moreover, this run will contain only finite paths. The analogous claims hold for partial (w, q_r)-runs and partial (w, q_{r^*})-runs.
2. Let $w \in \mathfrak{T}$ and let $q_v \in Q$. There exists an accepting partial (w, q_v)-run iff the tree $\mathfrak{T}^w$ rooted at w has a pre-match into $(\mathcal{G}, v)$.
3. Let $w \in \mathfrak{T}$ and let $q_{r^*v} \in Q$. There exists an accepting partial (w, q_{r^*v})-run iff the tree $\mathfrak{T}^w$ rooted at w has an r-reachable pre-match into $(\mathcal{G}, v)$.
4. Given $w \in \mathfrak{T}$ and $q \in Q$, let $\mathfrak{T}_{\mathsf{r}'}$ be a partial (w, q)-run and p an infinite path in $\mathfrak{T}_{\mathsf{r}'}$. If q occurs infinitely often on p, it is of the form q_{r^*v}.
5. If $\mathfrak{T}_{\mathsf{r}'}$ is an accepting partial (w, q)-run, all its paths are finite.

Claim 1 follows directly from the definition of the transition function $\delta(q_-, w)$, $\delta(q_r, w)$ and $\delta(q_{r^*}, w)$ in $\mathfrak{A}^{\mathcal{G}, v_0}$.

Claim 2 follows from the transition function $\delta(q_v, w)$ inductively as follows. For the right-to-left direction, we show the contrapositive: If there is no pre-match from $\mathfrak{T}^w$ rooted at w into $(\mathcal{G}, v)$, then the concept labels in $w.\mathbf{A}$ do not match the label of v in $\mathcal{G}$, and the run fails.

For the left-to-right direction, we assume there is a pre-match from $\mathfrak{T}^w$ into $(\mathcal{G}, v)$. In that case the labels in w match those in v. If v has no outgoing edges, then the automaton can choose to either search for a child node $w \cdot i$ with $w \cdot i.R = -$, or with $w \cdot i.R = r_\star$ and the concept labels in $w \cdot i.\mathbf{A}$ match the label of v. Since there exists a pre-match from $\mathfrak{T}^w$ into $(\mathcal{G}, v)$, one of these cases must be successful. In the former case, there exists a partial $(w \cdot i, q_-)$-run, which is finite by Claim 1, and therefore accepting. Similarly, if the latter check is successful, there exists an accepting partial $(w \cdot i, q_{r_\star})$-run, and since the labels in $w \cdot i.\mathbf{A}$ match the label of v, there is a pre-match from $w \cdot i$ into $(\mathcal{G}, v)$, and the same holds for any further successor nodes of $w \cdot i$ along the branch. Since we have defined $\mathfrak{T}$ as a proper node-labelled tree, every branch in $\mathfrak{T}^w$ will reach a node $w' = (-, \{\top\})$ in a finite number of steps. Thus, we repeat the previous check until we have found such a node w', at which point we must have $w'.R = -$, and therefore the partial (w', q_-)-run is accepting by Claim 1. It then follows by the definition of $\delta(w, q_v)$, that the partial (w, q_v)-run is accepting.

In case v does have some outgoing edge $r(v, v') \in E$, the automaton has two options, one of which must give an accepting run. In the first option, it may choose to transition into $(w \cdot i, q_r)$ and $(w \cdot i, q_{v'})$. Since there is a pre-match from $\mathfrak{T}^w$ into $(\mathcal{G}, v)$, we will have that $w \cdot i.R = r$ for some child node $w \cdot i$ of w and there is a pre-match from the tree rooted at $w \cdot i$ into $(\mathcal{G}, v')$. Therefore, the partial $(w \cdot i, q_r)$-run and the partial $(w \cdot i, q_{v'})$-run are both accepting, and therefore so is the partial (w, q_v)-run. If the former option check fails, it may transition into $(w \cdot i, q_{r^*})$ and $(w \cdot i, q_{r^*v'})$. Since there must exist a pre-match from the tree rooted at $w \cdot i$ into $(\mathcal{G}, v')$, by Claim 1 and Claim 3 both the partial $(w \cdot i, q_{r^*})$-run and the partial $(w \cdot i, q_{r^*v'})$-run will be accepting and therefore so is the partial (w, q_v)-run.

Claim 3 follows similarly, from the transition function $\delta(q_{r^*v}, w)$. For the right-to-left direction, assume there exists an accepting partial (w, q_{r^*v})-run $\mathfrak{T}_{\mathsf{r}'}$. Assume for contradiction that there is no r-reachable pre-match from $\mathfrak{T}^w$ into $(\mathcal{G}, v)$. Then, by construction of $\delta(q_{r^*v}, w)$, the check $(0, q_v)$ will fail, so we must check $(w, q_{r^*v'})$ for some outgoing edge $r(v, v') \in E$. This check will be repeated until there are no more outgoing edges in $\mathcal{G}$, in which case the run will fail, or it will loop infinitely often, in which case (w, q_{r^*v}) will occur infinitely often in this branch. Since such a state has $p(q_{r^*v}) = 1$ this also results in a non-accepting run.

For the left-to-right direction, we assume there exists an r-reachable pre-match from $\mathfrak{T}^w$ into $(\mathcal{G}, v)$. In case v has no outgoing edges in $\mathcal{G}$, we move into (w, q_v). Since there exists a pre-match from $\mathfrak{T}^w$ into some node $(\mathcal{G}, v')$ which is reachable via an r-path in $\mathcal{G}$ from v, but v has no outgoing r-edges, this r-path must be vacuous and the pre-match exists at v itself, then by Claim 2 the partial run rooted at (w, q_v) will be accepting.

If there does exist some outgoing edge $r(v, v') \in E$, then we may also move into $(w, q_{r^*v'})$. By assumption, there also exists an r-reachable pre-match from $\mathfrak{T}^w$ into $(\mathcal{G}, v')$. So we may repeat this check along the r-path in $\mathcal{G}$ until we have reached the node v^i such that there exists a pre-match from $\mathfrak{T}^w$ into v^i, at which point the partial run rooted at (w, q_{v^i}) will be accepting. Then the partial $(w, q_{r^*v^i})$-run will accept as well, and hence the partial (w, q_{r^*v})-run is accepting.

For Claim 4, assume we have a partial (w, q)-run and p an infinite path in $\mathfrak{T}_{\mathsf{r}'}$ on which q occurs infinitely often. By Claim 1, q cannot be of the form q_-, q_r or q_{r^*}. If q is of the form q_v, then from the proof of Claim 2 and the properness of $\mathfrak{T}_{\mathsf{r}'}$, it follows that we will always reach a state q_- in a finite number of steps. Thus q_v also cannot occur infinitely often. Therefore any q occurring infinitely often on p must be of the form q_{r^*v}.

Claim 5 follows as a corollary from Claim 4.

It then follows that $\mathfrak{T}_{\mathsf{r}}$ is an accepting run iff there exists a pre-match from $\mathfrak{T}$ into $(\mathcal{G}, v_0)$. □

Complementing and intersecting ATAs. Given a matching k-ATA $\mathfrak{A}^{\mathcal{G},v_0}$, its *complement automaton* $\overline{\mathfrak{A}^{\mathcal{G},v_0}}$ can be defined in the usual way, by dualising the transition function and changing the priority function. For more details on this construction, we refer interested readers to [23].

From Lemma 2 and the construction of the complement automaton, we directly get the following corollary.

Corollary 1. *Given a data graph $\mathcal{G} = (V, E, \ell)$, a node $v_0 \in V$, and a k-ATA $\mathfrak{A}^{\mathcal{G},v_0}$, its complement automaton $\overline{\mathfrak{A}^{\mathcal{G},v_0}}$ accepts exactly those proper node-labelled k-trees $\mathfrak{T}$ such that there is not a pre-match from $\mathfrak{T}$ into v.*

Given a set of k-ATAs, $\mathscr{A}_n = \mathfrak{A}_1, \ldots, \mathfrak{A}_n$ of size n, where for each $\mathfrak{A}_i = \langle Q_i, \mathbb{L}_\Sigma, \delta_i, q_i, p_i \rangle$ with $1 \leq i \leq n$, we can define the intersection automaton as a k-ATA $\mathfrak{A}_{\mathscr{A}_n} = \langle Q_{\mathscr{A}_n}, \mathbb{L}_\Sigma, \delta_{\mathscr{A}_n}, q_{\mathscr{A}_n}, p_{\mathscr{A}_n} \rangle$ as follows:

$$Q_{\mathscr{A}_n} = \bigcup_{1 \leq i \leq n} Q_i \cup \{q_{\mathscr{A}_n}\}$$

$$\delta_{\mathscr{A}_n}(q, w) = \begin{cases} \bigwedge_{1 \leq i \leq n} (0, q_i) & \text{if } q = q_{\mathscr{A}_n} \\ \delta_i(q, w) & \text{if otherwise } q \in Q_i \text{ for some } i \end{cases}$$

$$p(q) = \begin{cases} 0 & \text{if } q = q_{\mathscr{A}_n} \\ p_i(q) & \text{if otherwise } q \in Q_i \text{ for some } i \end{cases}$$

We say that $\mathfrak{A}_{\mathscr{A}_n}$ is the intersection over $\mathscr{A}_n$.

Lemma 10. *Given a set of k-ATAs $\mathscr{A}_n = \{\mathfrak{A}_1, \ldots, \mathfrak{A}_n\}$, let $\mathfrak{A}_{\mathscr{A}_n}$ be the intersection over $\mathscr{A}$, then we have that $w \in L(\mathfrak{A}_{\mathscr{A}_n})$ iff $w \in \bigcap_{1 \leq i \leq n} L(\mathfrak{A}_i)$.*

Lemma 3. *Given an instance $\mathcal{F} = (\mathcal{G}, \mathscr{P}, \mathscr{N}, \Sigma)$, where $\mathcal{G} = (V, E, \ell)$, if there is an $\mathcal{EL}^*$ fitting for $\mathcal{F}$, then there is an $\mathcal{EL}^*$ fitting C such that $deg(\mathcal{T}_C) = |V|$.*

Proof. Take any $\mathcal{EL}^*$ fitting C for $\mathcal{F}$. Let $\mathcal{T}_C$ be the $\mathcal{EL}^*$ tree with degree k for C, where $k > |V|$. Let u_0 be the root node of $\mathcal{T}_C$. Let u by any node in $\mathcal{T}_C$. By construction we know that u has exactly k successors, $y_1, \ldots, y_k$. For $1 \leq i \leq k$, let $\mathcal{T}_C|_u^i$ be the subtrees rooted at u, obtained by deleting all successors $y_i, \ldots, y_k$ and the subtrees below them from u. Let S_i be the set of all $v \in V$ such that there is a pre-match ρ from $\mathcal{T}_C|_u^i$ to $\mathcal{G}$ with $\rho(u) = v$. From this definition, we have $S_1 \supseteq S_2 \supseteq \ldots$. There must then exist some $j \leq |V|$, such that $S_j = S_{j+1} = \cdots = S_k$. This follows from the fact that $k > |V|$ and hence we cannot decrease the sets S_i more often than there are elements in V.

Let $\mathcal{T}_D$ be the $\mathcal{EL}^*$ tree obtained from replacing u in $\mathcal{T}_C$ with $\mathcal{T}_D|_u^j$ and D the corresponding concept of $\mathcal{T}_D$, i.e. of the subtree we get by removing all successors $y_{j+1}, \ldots, y_k$ of u from $\mathcal{T}_C$. We show that D is a fitting for $\mathcal{F}$. It follows from our construction that D fits all the positive examples, since we obtained D by dropping subexpressions from C. It remains to show that D fits none of the elements in $\mathscr{N}$, or formally we want to show that there does not exist an $n \in \mathscr{N}$ s.t. $n \in D^{\mathcal{G}}$. Let us assume to the contrary that there is some $n \in \mathscr{N}$ such that there is a pre-match ρ from $\mathcal{T}_D$ to $\mathcal{G}$, where

$\rho(u_0) = n$ (and therefore also $n \in D^{\mathcal{G}}$ via Lemma 1). Then it follows that we can construct a pre-match ρ' from $\mathcal{T}_C$ to $\mathcal{G}$, with $\rho'(u_0) = n$. To see why, consider that the only difference between $\mathcal{T}_D$ and $\mathcal{T}_C$ lies in the successors $y_{j+i}, \ldots, y_m$ that we removed in $\mathcal{T}_D$. Yet we know that $S_j = S_{j+1} = \cdots = S_k$. In other words, we know that for all elements in S_j, there must be pre-matches to extend them to $\mathcal{T}_C$. This, however, contradicts our assumption that C was a fitting for $\mathcal{F}$.

Note that this construction only reduces the branching of *one* node to at most j, where $j \leq |V|$.

By applying this construction for any node in $\mathcal{T}_C$, we can construct a $\mathcal{EL}^*$ tree with degree $k' = |V|$ such that its corresponding concept is a fitting for $\mathcal{F}$. □

Theorem 1. *Given a fitting instance $\mathcal{F} = (\mathcal{G}, \mathscr{P}, \mathscr{N}, \Sigma)$, let k be the number of nodes in $\mathcal{G}$, and let $\mathfrak{A}_{\mathcal{F}}$ be the k-ATA obtained by intersecting $\mathfrak{A}^{\mathcal{G},v}$ for every $v \in \mathscr{P}$ and $\overline{\mathfrak{A}^{\mathcal{G},n}}$ for each $n \in \mathscr{N}$. Then $\mathfrak{A}_{\mathcal{F}}$ accepts a proper node-labelled k-tree $\mathfrak{T}_C$ over the signature Σ if and only if C is an $\mathcal{EL}^*$ fitting for $\mathcal{F}$.*

Proof. We show the claim by considering each direction separately.
For the left-to-right direction, we assume that the proper node-labelled $\mathcal{EL}^*$ tree $\mathfrak{T}_C$ is accepted by $\mathfrak{A}_{\mathcal{F}}$ and let C be its corresponding $\mathcal{EL}^*$ concept. We know from Lemma 2 that each $\mathfrak{A}^{\mathcal{G},v}$ accepts proper $\mathcal{EL}^*$ k-trees when they have a pre-match into $v \in \mathscr{P}$. Furthermore, by Corollary 1, we know that each $\overline{\mathfrak{A}^{\mathcal{G},n}}$ only accepts proper $\mathcal{EL}^*$ k-trees if they have no pre-match into any $n \in \mathscr{N}$. From Lemma 10, we know that $\mathfrak{T}_C$ is accepted by every automaton in $\mathscr{A}_m$, so also by the intersection automaton $\mathfrak{A}_{\mathcal{F}}$ over $\mathscr{A}_m$ and together with Lemma 1 it follows that for every $v \in \mathscr{P}$, we have $v \in C^{\mathcal{G}}$ and for every $n \in \mathscr{N}$, we have $n \notin C^{\mathcal{G}}$. Combined with the observation that $\Sigma(C) \subseteq \Sigma$ by our initial assumption, it follows that C is a fitting for $\mathcal{F}$.

For the right-to-left direction, we assume that there exists some $\mathcal{EL}^*$ concept C that is a fitting for $\mathcal{F} = (\mathcal{G}, \mathscr{P}, \mathscr{N}, \Sigma)$. From Lemma 3 it follows that there must exist some D such that its proper node-labelled tree encoding $\mathfrak{T}_D$ has degree $\leq |V|$. We know that for every $v \in \mathscr{P}$ (resp. every $n \in \mathscr{N}$), it holds that $v \in D^{\mathcal{G}}$ (resp. $n \notin D^{\mathcal{G}}$). By Lemma 1, it follows that $\mathfrak{T}_D$ has a pre-match into each $v \in \mathscr{P}$ (resp. has no pre-match into any $n \in \mathscr{N}$). Combining this observation with Lemma 2 and its corollary, we know that $\mathfrak{T}_D$ is accepted by every automaton in $\mathscr{A}_k$, if we set $k = |V|$. From Lemma 10 it then follows that $\mathfrak{T}_D$ is accepted by the intersection automaton $\mathfrak{A}_{\mathcal{F}}$ over $\mathscr{A}_k$. □

B. Proofs for Section 5 (Computation of Most Specific Fittings)

Pre-matches, homomorphisms and simulations In addition to the pre-matches and $\mathcal{EL}^*$ simulations introduced in the body of the paper, for some proofs we also need to use the more standard notions of homomorphisms and simulations as in the $\mathcal{EL}$ setting.

First, we extend our notion of pre-matches from Section 4 so that we may use starred graphs on both sides, i.e. we change condition 3 as follows:

$$\text{if } (v, r_\star, v') \in E, \qquad \text{then } (\rho(v), \rho(v')) \in ((r \cup r_\star)^*)^{\mathcal{G}} .$$

Then a *homomorphism* is defined as a pre-match that does not use condition 3 of Definition 4. This corresponds to the regular definition of homomorphisms, but where $r_\star$ can be used and is treated the same as a regular symbol r. Note that homomorphisms have the following property, which ensures that paths in unravellings that are homomorphic are of the same length:

> (‡) Given two pointed tree-shaped graphs $(\mathcal{I}, d), (\mathcal{G}, e)$ such that there exists a homomorphism $h : (\mathcal{I}, d) \to (\mathcal{G}, e)$, then for any path $p \in \mathcal{I}$, $length(p) = length(h(p))$.

Moreover, we define a $\mathcal{EL}$ Σ-*simulation* as a less restricted version of the $\mathcal{EL}^*$ Σ-simulation introduced in Definition 9, which does not require the third condition **Forth***. We write $(\mathcal{I}, d) \preceq_\Sigma^{\mathcal{EL}} (\mathcal{G}, e)$ to denote that there exists an $\mathcal{EL}$ Σ-simulation between $(\mathcal{I}, d)$ and $(\mathcal{G}, e)$ w.r.t. some signature Σ. Like in the case of $\mathcal{EL}^*$ Σ-simulations, when $\Sigma = \Sigma(\mathcal{I})$ we may omit it and simply write $(\mathcal{I}, d) \preceq^{\mathcal{EL}} (\mathcal{G}, e)$. Note that if an $\mathcal{EL}$ Σ-simulation exists between two non-starred graphs, then we have an analogous version of Lemma 6 for $\mathcal{EL}$ concepts.

In the case of tree-shaped graphs, it is well-known that an $\mathcal{EL}$ simulation between two graphs exists iff there exists a homomorphism. The same result holds for our notions of $\mathcal{EL}^*$ simulations and pre-matches.

Lemma 9. *For all Σ and all $(\mathcal{I}, d)$ and $(\mathcal{G}, e)$ such that $\mathcal{G}$ is $\star$-closed,*

1. *For every $\mathcal{EL}^*$ concept C with $\Sigma(C) \subseteq \Sigma$ s.t. $\mathcal{T}_C \preceq \mathcal{U}_e$, we have $\mathcal{T}_C \preceq \mathcal{U}_e^m$, where $m = depth(C)$.*
2. *If $\mathcal{I}$ and $\mathcal{G}$ are finite, then $\mathcal{U}_d \preceq_\Sigma (\mathcal{G}, e)$ iff $\mathcal{U}_d^m \preceq_\Sigma (\mathcal{G}, e)$ for all m-unravellings $\mathcal{U}_d^m$ of $\mathcal{I}$ at d.*

Proof. To prove Item 1 of Lemma 9, We make use of the following claim:

> (†) If $\mathcal{G}$ is a $\star$-closed graph with some node v, then for every $\mathcal{EL}^*$ concept C, there exists a pre-match ρ from $\mathcal{T}_C$ into $\mathcal{U}_v$, iff there exists a regular homomorphism $h : \mathcal{T}_C \to \mathcal{U}_v$.

The right-to-left direction follows immediately, since a homomorphism in our setting can be seen as a special case of pre-matches where for any r or $r_\star$-edge in the description tree, there exists an r or $r_\star$-edge in the graph.

The left-to-right direction of the claim follows from the following two properties of $\mathcal{U}_v$:

1. For any two nodes p, p' in $\mathcal{U}_v$, if $tail(p) = tail(p')$, then the subtrees below p, p' are isomorphic to each other.
2. By the fact that $\mathcal{G}$ is $\star$-closed, it follows that for any node p in $\mathcal{U}_v$, if there exists a node q which can be reached from p via an $(r \cup r_\star)$-path, then there exists a node q' which can be reached from p via a single $r_\star$-edge s.t. $tail(q) = tail(q')$.

Then given a $\star$-closed graph $\mathcal{G}$ with some node v and some C s.t. $\mathcal{T}_C = (V, E, \ell)$ has a pre-match ρ into $\mathcal{U}_v$ such that

$$(v, r_\star, v') \in E \text{ and } (\rho(v), \rho(v')) \in ((r \cup r_\star)^*)^{\mathcal{U}_v} \text{ and } (\rho(v), \rho(v')) \notin (r \cup r_\star)^{\mathcal{U}_v},$$

and by the fact that $\mathcal{G}$ is $\star$-closed there exists a pre-match ρ' from $\mathcal{T}_C$ into $\mathcal{U}_v$ such that

$$(v, r_\star, v') \in E \text{ and } (\rho'(v), \rho'(v')) \in (r_\star)^{\mathcal{U}_v},$$

with $\rho(v) = \rho'(v)$ and $tail(\rho(v')) = tail(\rho'(v'))$. Then the subtrees of $\rho(v')$ and $\rho'(v')$ are isomorphic, and therefore ρ' is a homomorphism $\rho' : \mathcal{T}_C \to \mathcal{U}_v$.

To prove Lemma 9 Item 1, we assume that given a $\star$-closed graph $\mathcal{G}$ with node v and an $\mathcal{EL}^*$ concept C with $\Sigma(C) \subseteq \Sigma$, we have $\mathcal{T}_C \preceq \mathcal{U}_v$. Then there also exists a pre-match from $\mathcal{T}_C$ into $\mathcal{U}_v$. It then follows from (†) that there exists a homomorphism from $\mathcal{T}_C$ to $\mathcal{U}_v$. Let $m = depth(C)$. It follows from the property (‡) for homomorphisms that there also exists a homomorphism from $\mathcal{T}_C$ to $\mathcal{U}_v^m$. By the other direction of (†), there is then also a pre-match from $\mathcal{T}_C$ to $\mathcal{U}_v^m$ and hence also $\mathcal{T}_C \preceq \mathcal{U}_v^m$.

□

To prove Theorem 3, we make use of the following claim:

(††) Given a graph $\mathcal{G}$ with a node v. Let $\mathcal{U}_v$ be the full unravelling of $\mathcal{G}$ at v and let the longest path in $\mathcal{U}_v$ be of size n. If $\mathcal{U}_v^m$ is the strict m-bounded unravelling of $\mathcal{G}$ at v for some $m < n$, then $\mathcal{U}_v \not\preceq \mathcal{U}_v^m$.

Proof. Given a graph $\mathcal{G}$ with a node v its unravelling $\mathcal{U}_v$ with the longest path of size n, and the bounded unravellings $\mathcal{U}_v^m, \mathcal{U}_v^{m'}$ with $m, m' \geq 0$, assume there exists a homomorphism $h : \mathcal{U}_v^m \to \mathcal{U}_v^{m'}$. We use the claim (‡) that for every path $p \in \mathcal{U}_{\vec{v}_+}^m$, $length(p) = length(h(p))$. Then it must be the case that $m' \geq m$, meaning that a unravelling can only have a homomorphism into a larger unravelling of the same graph, and never into one that is strictly smaller. It follows that the full unravelling $\mathcal{U}_v$ cannot have a homomorphism into any strict bounded unravelling of the same graph, since any path in a strict bounded unravelling is shorter than n. Thus $\mathcal{U}_v \not\twoheadrightarrow \mathcal{U}_v^m$ for any $m < n$. Since the unravellings are tree-shaped, since there exists no homomorphism, then there also exists no $\mathcal{EL}$ simulation, i.e. $\mathcal{U}_v \not\preceq^{\mathcal{EL}} \mathcal{U}_v^m$. Therefore, since $\mathcal{EL}$ simulations are a less restricted version of $\mathcal{EL}^*$ simulations, we also have $\mathcal{U}_v \not\preceq \mathcal{U}_v^m$. □