Towards Model Transformation in Description Logics – Investigating the Case of Transductions

Willi Hieke and Anni-Yasmin Turhan *

TU Dresden, Dresden, Germany
Institute of Theoretical Computer Science
{willi.hieke, anniyasmin-turhan}@tu-dresden.de

Abstract. Models for Description Logic (DL) ontologies can be used for explanation purposes, but the models computed by standard DL reasoner systems are not necessarily suitable for this task. In this paper we investigate the general task of transforming an arbitrary model into a target model that admits the desired property. To this end, we introduce a general framework for model transformation that abstracts from the DL in use, the property of the target model and the transformation formalism. We consider instantiations of the framework for the DLs $\mathcal{ALC}$ and $\mathcal{EL}$ and use transductions as transformation formalism.

Keywords: Description logics · Model transformation · Transduction.

1 Introduction

Description Logics (DLs) are a family of knowledge representation languages that are widely used in ontology-based applications. Most DLs are fragments of first-order logic (FOL) that have decidable reasoning problems. Ontologies usually capture definitions for terminologies from an application domain by concept axioms. These axioms can refer to other concepts or roles, which are unary and binary predicates, respectively.

DL ontologies can easily grow very large and complex and thus, a consequence computed by a DL reasoner is not necessarily obvious for a knowledge engineer and automated ontology services to explain the results of reasoning are needed. Such a service is the computation of justifications and was intensively investigated in recent years, see [10] for a recent overview. Using justifications and proofs for explanation identifies parts of an ontology that is "responsible" for a consequence, whereas the approach for explanation presented in this paper uses models. For instance, in the life-cycle of an ontology, the developer team might change, so to acquire the meaning of concepts and to grasp the overall structure of an ontology might not be an easy task for a new knowledge engineer. What could be helpful is a tool for the computation of typical instances that are easy to comprehend, but still carry the information from the ontology.

* This work has been supported by DFG in the Research Training Group QuantLA (GRK 1763).

For a range of DLs there are highly-optimized reasoner systems available such as FaCT++ [14], HermiT [8], or ELK [9]. DL reasoners decide satisfiability of an ontology (or a concept) by computing a model of it. However, using such a model for explanation purposes may not be appropriate, since it is simply the first model found according to the reasoner's optimization strategies and may be artificial and counter-intuitive. It is a natural idea to transform a model such that it would be better suited for explanation. Explorations of this idea in the context of DLs are scarce. In the work of Bauer et al. [3], where the internals of the FaCT++ reasoner [14] have been modified to guide the search of a model s.t. a model (more) suitable for explanation is obtained. As changing the internals of a highly-optimized reasoner can easily corrupt the implementation, the approach where reasoner generated models are transformed is clearly preferable. In [1] the authors adopt this approach and suggest visualizations of concepts by (classes of) models, where information "not relevant to" the user's view is filtered out. The general task of model transformation has been investigated in the more general setting of FOL formulae. For instance, computing minimal models has been investigated in [4]. However, as for FOL reasoning is not decidable and transformations for these models need to cater for n-ary relations, these methods need not be suitable for the DL case. Similarly, there is a lot of work on model transformation for propositional formulae, but these lack the relational structures that DL models have and therefore these methods are not applicable.

As classical DLs use only unary predicates and binary relations, their models are node and edge labeled graphs and can be easily be depicted. For a graphical representation of a concept instance, it could be advantageous, if the model was for instance small, or tree-like, or planar, and so on. We consider these properties here—not so much to argue for their cognitive adequateness for explanation, but rather to illustrate by their use that our approach can handle properties that can reasonably be required for these purposes. We address the task of model transformation by introducing a framework, in which the intended property to be satisfied by the model is kept variable. This allows for different properties depending on the application. We introduce a general framework for transforming arbitrary models of ontologies. This framework is subsequently used to formalize reasoning problems and to investigate a prominent one in depth in this paper.

As the formalisms for transformations, we use *transductions* (as defined in [7]) in this paper. Transductions specify mappings on relational structures using logical formulae with free variables. Transductions are a promising tool for transformations as they admit the expressivity of monadic second order logic in their formulae. However, in principle our framework admits the use of other graph transformation formalisms, as for instance Graph Rewriting Systems (GRS) [11].

As a concrete instantiation of our framework, we investigate the case of transforming arbitrary models of ontologies written in the DL called $\mathcal{ALC}$ into a tree(-like) models by transductions. We construct a family of transductions over interpretations and show that these transformations are model preserving, i.e. input and output model satisfy the same sentences in the DL $\mathcal{ALC}$. We also briefly consider a lightweight DL that is of limited expressiveness, but has good com-

putational properties and has the canonical model property (sometimes called universal model). The canonical model would allow for imposing assumptions on the model that is to be transformed.

This paper is structured as follows: in Section 2, we define the basic notions of DL and transductions. We define the model transformation framework and dedicated reasoning problems for it in Section 3 and consider the case of transductions to tree-like models in Section 4 as an instantiation of our framework for $\mathcal{ALC}$ knowledge bases. We also discuss aspects of model transformation for $\mathcal{EL}$. Last, we draw conclusions and lay out future work in Section 5.

2 Preliminaries

As this paper mainly combines notions from Description Logics and graph transductions, we briefly introduce the main notions from these fields. For detailed introductions, see [2] or [7], respectively.

2.1 Description Logics

The main building blocks for DL knowledge bases are concepts, which describe unary predicates from the modeled domain. We use sets for concept names $\mathsf{N_C}$ and for role names $\mathsf{N_R}$. Let $A \in \mathsf{N_C}$ and $r \in \mathsf{N_R}$, then *complex $\mathcal{ALC}$ concepts* can be built according to the following grammar:

$$C := A \mid \top \mid \neg C \mid C \sqcap C \mid \exists r.C \ .$$

Commonly used abbreviations are: $C_1 \sqcup C_2 := \neg(\neg C_1 \sqcap \neg C_2)$, $\bot := \neg\top$, and $\forall r.C := \neg(\exists r.\neg C)$. The semantics is defined as in first-order logic by means of *interpretations*, which are a tuple $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$, consisting of the domain $\Delta^\mathcal{I}$ and a function $\cdot^\mathcal{I}$ that maps elements from $\mathsf{N_C}$ to subsets of the domain and elements from $\mathsf{N_R}$ to subsets over $\Delta^\mathcal{I} \times \Delta^\mathcal{I}$. The mapping $\cdot^\mathcal{I}$ is extended to (complex) $\mathcal{ALC}$ concept descriptions as follows: $\top^\mathcal{I} = \Delta^\mathcal{I}$, $\neg C^\mathcal{I} = \Delta^\mathcal{I} \setminus C^\mathcal{I}$, $(C_1 \sqcap C_2)^\mathcal{I} = C_1^\mathcal{I} \sqcap C_2^\mathcal{I}$ and $(\exists r.C)^\mathcal{I} = \{d \in \Delta^\mathcal{I} \mid \exists e.(d,e) \in r^\mathcal{I} \text{ and } e \in C^\mathcal{I}\}$. The fragment of $\mathcal{ALC}$ that only uses the top-concept $\top$, conjunction ($\sqcap$), and existential restrictions ($\exists r.C$) as concept constructors is called $\mathcal{EL}$. Note that $\mathcal{EL}$ cannot express inconsistencies.

Terminological knowledge about the application domain can be modeled by using (complex) concept descriptions. Let C and D be concepts, a *general concept inclusion* (GCI) is a statement of the form: $C \sqsubseteq D$. A *TBox* $\mathcal{T}$ is a finite set of GCIs. An interpretation $\mathcal{I}$ *satisfies a GCI* $C \sqsubseteq D$, if $C^\mathcal{I} \subseteq D^\mathcal{I}$. An interpretation $\mathcal{I}$ is a *model of a TBox* ($\mathcal{I} \models \mathcal{T}$) if and only if $\mathcal{I}$ satisfies $C \sqsubseteq D$ for all $C \sqsubseteq D \in \mathcal{T}$. A *knowledge base* $\mathcal{K}$ is a pair of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$. The latter can express knowledge about objects by the use of constants. We concentrate on terminological knowledge and do not introduce ABoxes formally. We assume $\mathcal{A}$ to be empty in $\mathcal{K}$. We denote by $\Sigma(\mathcal{I})$ the *signature* of $\mathcal{I}$, i.e., the set of concept and role names occurring in $\mathcal{I}$; and for concepts C and GCIs,

respectively. We write $\Sigma_{\mathsf{A}} := \Sigma \cap \mathsf{N_C}$ and $\Sigma_{\mathsf{r}} := \Sigma \cap \mathsf{N_R}$ and we assume signatures to be finite.

Typical reasoning problems for DLs are to decide *satisfiability* and *subsumption*. *Satisfiability* checks for the existence of a model for a given concept or TBox and *subsumption* checks for super-concept relationships between two given concepts w.r.t. a TBox and is formally defined as: D *subsumes* C w.r.t. TBox $\mathcal{T}$ (written $C \sqsubseteq_{\mathcal{T}} D$) iff in all models $\mathcal{I}$ of $\mathcal{T}$ holds: $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.

DLs differ in their expressivity and this often shows in the complexity for reasoning. For $\mathcal{ALC}$, testing subsumption of concepts w.r.t. a non-empty TBox is ExpTime-complete [12] and w.r.t. for an empty one it is PSpace-complete [13]. In $\mathcal{EL}$ deciding subsumption can already be done in polynomial time [6]. The expressivity of a DL can be captured by invariance results. Let $\mathcal{L}$ be a logic and $\bowtie$ be a relation between two pointed interpretations, which are an interpretation together with a dedicated element form its domain. An invariance result for $\mathcal{L}$ then says, that for all pointed interpretations $(\mathcal{I}, d)$ and $(\mathcal{J}, e)$ with $(\mathcal{I}, d) \bowtie (\mathcal{J}, e)$ and an $\mathcal{L}$-formula ϕ: $\mathcal{I} \models \phi(d)$ holds iff $\mathcal{J} \models \phi(e)$ holds. In case of $\mathcal{ALC}$ such an invariance result holds for bisimulations [5], while for $\mathcal{EL}$ it holds for homomorphisms.

2.2 Transductions

We now define transductions in regard to DL interpretations. Observe that such an interpretation is effectively a directed graph with labeled vertices (indicating concept membership) and labeled edges (indicating membership to a role). Intuitively in a transduction, vertices and edges from one interpretation are being copied into a new interpretation if they satisfy a condition given by a logical formula. Depending on the fulfillment of the conditions, the new copy can be modified. If the conditions are formulated in monadic second-order logic (MSO) or FOL, we call the transduction a MSO- or FOL-transduction, respectively.

Since typical DL interpretations have at most binary predicates, we restrict the standard definition of transductions of relational structures from [7] to the case of at most binary relations. We denote the set of $\mathcal{L}$-formulae (e.g. $\mathcal{L} \in \{\text{MSO, FOL}\}$) over Σ, with free variables in X, by $\mathcal{L}(\Sigma, X)$. The set of all Σ-interpretations is denoted by $\mathfrak{I}(\Sigma)$.

Let Σ and Σ' for the remainder of this paper be binary signatures. A *transduction* τ of type from Σ to Σ' is a binary relation on Σ-interpretations or formally, $\tau \subseteq \mathfrak{I}(\Sigma) \times \mathfrak{I}(\Sigma')$. We call the input of a transduction *source interpretation* and its output *target interpretation*. The logical formulae that capture the transformation conditions and thereby induce a particular transduction are comprised in a tuple called *definition scheme*.

Definition 1 (Definition scheme). *Let* Par *be a finite set of variables called* parameters. *Let* AuxVar *be a finite set of variables called* auxiliary variables *s.t.* $\mathsf{Par} \cap \mathsf{AuxVar} = \emptyset$, *let* $x, y \in \mathsf{AuxVar}$, *and* $\mathcal{L}$ *be a logic. A* definition scheme *of type from* Σ *to* Σ' *with set of parameters* Par *is a tuple of formulae of the form*

$$\mathsf{D} = (\chi, (\delta_i)_{i \in [k]}, (\theta_\omega)_{\omega \in (\Sigma'_{\mathsf{A}} \times [k])}, (\eta_\omega)_{\omega \in (\Sigma'_{\mathsf{r}} \times [k]^2)})$$

for some $k \in \mathbb{N}^+$ *(we call positive integers* $\mathbb{N}^+$ index set *and abbreviate* $\{1, \ldots, k\}$ *with* $[k]$*), where*

- $\chi \in \mathcal{L}(\Sigma, \mathsf{Par})$ *is the* precondition*;*
- $\delta_i \in \mathcal{L}(\Sigma, \mathsf{Par} \cup \{x\})$ *for each* $i \in [k]$ *are* domain formulae*;*
- $\theta_\omega \in \mathcal{L}(\Sigma, \mathsf{Par} \cup \{x\})$ *for each* $\omega \in \Sigma'_{\mathsf{A}} \times [k]$*, and are* concept formulae*; and*
- $\eta_\omega \in \mathcal{L}(\Sigma, \mathsf{Par} \cup \{x, y\})$ *for each* $\omega \in \Sigma'_{\mathsf{r}} \times [k]^2$ *are the* role formulae*.*

A definition scheme D *is an* $\mathcal{L}$-definition scheme*, if all its formulae are written in some logic* $\mathcal{L}$*. If* $\mathsf{Par} = \emptyset$ *for* D*, then* D *is called* parameterless*.*

Please note that a definition scheme is also called a *graph transducer* in [15]. Intuitively, any input interpretation has to satisfy the precondition χ first, before the domain formulae δ_i and the relation formulae θ_ω and η_ω are being evaluated on the input interpretation. The domain formulae then select the elements from the input interpretation that satisfy δ_i. By using not only one but many domain formulae, elements can be copied more than once into target interpretation. Here, k is the upper bound on the number of copies of each domain element from the source interpretation. Consequently, the domain of the target interpretation consists of at most k distinct copies of the input domain. The relation formulae θ_ω and η_ω then state the conditions under which a concept name or a role name is added to the copied elements.

Next, we define how a target interpretation is induced by a definition scheme from a source interpretation $\mathcal{I}$. If applied to sets of interpretations and parameter assignments, this method then gives rise to a transduction.

Definition 2 (Transduction induced by a definition scheme). *Let* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ *be a* Σ*-interpretation,* $\lambda_{\mathsf{Par}} : \mathsf{Par} \to \Delta^{\mathcal{I}}$*, be a parameter assignment, and*

$$\mathsf{D} = (\chi, (\delta_i)_{i \in [k]}, (\theta_\omega)_{\omega \in (\Sigma_{\mathsf{A}} \times [k])}, (\eta_\omega)_{\omega \in (\Sigma_{\mathsf{r}} \times [k]^2)})$$

a definition scheme. Let λ *be the assignment that extends* λ_{Par} *s.t.* $\lambda(x) = a$ *and* $(\mathcal{I}, \lambda_{\mathsf{Par}}) \models \delta_i(a)$ *stand for* $(\mathcal{I}, \lambda) \models \delta_i$ *(and extending it accordingly for the* θ_i *and* η_i*). Then* $\mathcal{I}'$ *is the* Σ'-interpretation defined by D *from* $(\mathcal{I}, \lambda_{\mathsf{Par}})$ *iff:*

- $(\mathcal{I}, \lambda_{\mathsf{Par}}) \models \chi$*,*
- $\Delta^{\mathcal{I}'} := \{(a, i) \in \Delta^{\mathcal{I}} \times [k] \mid (\mathcal{I}, \lambda_{\mathsf{Par}}) \models \delta_i(a)\}$*,*
- $\mathsf{A}^{\mathcal{I}'} := \{(a, i) \in \Delta^{\mathcal{I}'} \mid (\mathcal{I}, \lambda_{\mathsf{Par}}) \models \theta_\omega(a)\}$ *for all* $A \in \Sigma_{\mathsf{A}}(\mathcal{I})$*, and*
- $\mathsf{r}^{\mathcal{I}'} := \{((a_1, i_1), (a_2, i_2)) \in (\Delta^{\mathcal{I}'} \times \Delta^{\mathcal{I}'}) \mid (\mathcal{I}, \lambda_{\mathsf{Par}}) \models \eta_\omega(a_1, a_2)\}$ *for all* $r \in \Sigma_{\mathsf{r}}(\mathcal{I})$*.*

As the interpretation $\mathcal{I}'$ *is uniquely determined by* D*,* $\mathcal{I}$*, and* λ *iff* $(\mathcal{I}, \lambda) \models \chi$*, we obtain the* transduction τ_{D} induced by D *and use the function* $\tau_{\mathsf{D}}(\mathcal{I}, \lambda) = \mathcal{I}'$ *directly.*

Note that transductions, as in [7], are defined over finite structures and we restrict ourselves to finite interpretations accordingly. This does not pose a restriction for our goal to transform models that are computed by a DL reasoner as these are finite.

3 A General Framework for Model Transformation

To be able to investigate model transformations in a wider setting in a structured and well-defined way, we define a general framework that abstracts from the concrete property to be achieved for the target interpretation, the knowledge base, and even the logics employed. The parameters of the framework are, the knowledge base $\mathcal{K}$ under consideration together with an interpretation $\mathcal{I}$ over the same signature. We write $\mathcal{L}_\mathcal{K}$ for the logic in which the knowledge base is formulated. The desired property is a sentence ρ of some logic $\mathcal{L}_\rho$. We sometimes write ρ-model for a model that satisfies property ρ. Another parameter is the actual transformation mapping. As we concentrate on transductions, we use τ to transform arbitrary source models of $\mathcal{K}$ to desired ρ-models.

Definition 3 (Model transformation framework). *Let $\mathcal{L}_\mathcal{K}$ be a description logic, $\mathcal{K}$ an $\mathcal{L}_\mathcal{K}$-knowledge base, $\mathcal{I}$ an interpretation over the signature $\Sigma(\mathcal{K})$, ρ a sentence formulated in some logic $\mathcal{L}_\rho$ and $\tau : \mathfrak{I}(\Sigma(\mathcal{K})) \to \mathfrak{I}(\Sigma(\mathcal{K}))$ a transformation mapping over interpretations of the binary signature $\Sigma(\mathcal{K})$. We call $S = (\mathcal{I}, \mathcal{K})$ the* source pair *and $T = (\rho, \tau)$ the* transformation pair*.*

We use transductions to describe model transformations and restrict ourselves to empty ABoxes and finite interpretations. In principle, the framework would also allow for other kinds of transformations or even structures of higher arity. The main reasoning tasks using the framework is to decide if a transformation pair applied to a source pair is successful.

Definition 4 (Successfulness problems). *Let $S = (\mathcal{I}, \mathcal{K})$ be a source pair and $T = (\rho, \tau)$ be a transformation pair. Let $\mathcal{L}_\mathcal{K}$ be a DL. A pair (S, T), is called* successful*, denoted by* $\mathsf{successful}(S, T)$*, iff $\mathcal{I} \models \mathcal{K}$ implies $\tau(\mathcal{I}) \models \mathcal{K} \cup \{\rho\}$.*

- *The* successfulness problem for S and T *is to decide for a given (S, T) with $S = (\mathcal{I}, \mathcal{K})$ and $T = (\rho, \tau)$, whether (S, T) is successful.*
- *The* successfulness problem for a DL and a transformation pair *is to decide for a given DL $\mathcal{L}_\mathcal{K}$ and $T = (\rho, \tau)$, whether for every $S = (\mathcal{I}, \mathcal{K})$, where $\mathcal{K}$ is expressed in $\mathcal{L}_\mathcal{K}$, the pair (S, T) is successful.*

Please note that, in case $\mathcal{I} \not\models \mathcal{K}$, the success of a pair (S, T) is trivially given. Furthermore, $\mathcal{K}$ and ρ can be inconsistent, meaning that there is simply no model for $\mathcal{K} \cup \rho$. Thus the successfulness property does not hold trivially in every case.

In principle the framework can easily give rise to other reasoning problems. An interesting question that generalizes the successfulness problem is whether for a given DL $\mathcal{L}_\mathcal{K}$ and a given property ρ, a $\mathcal{L}_\mathcal{K}$-KB has a ρ-model.

Definition 5 (ρ-model existence problems). *Given a DL $\mathcal{L}_\mathcal{K}$ and a property ρ formulated in a logic $\mathcal{L}_\rho$.*

- *The* ρ-model existence problem for $\mathcal{K}$ *decides whether there exists a ρ-model for a given KB $\mathcal{K}$ formulated in $\mathcal{L}_\mathcal{K}$.*
- *The* ρ-model existence problem for $\mathcal{L}_\mathcal{K}$ *decides whether there exists a ρ-model for every consistent KB $\mathcal{K}$ formulated in $\mathcal{L}_\mathcal{K}$.*

To answer these question may necessitate to compare the expressiveness of $\mathcal{L}_\mathcal{K}$ and the logic $\mathcal{L}_\rho$ that ρ is formulated in. To decide the ρ-model existence problem for a given KB $\mathcal{K}$ where $\mathcal{L}_\rho$ is not more expressive than $\mathcal{L}_\mathcal{K}$, the condition to fulfill the property ρ can, in principle be answered by employing the DL reasoner for $\mathcal{L}_\mathcal{K}$. For some cases, the ρ-model existence problem for a DL $\mathcal{L}_\mathcal{K}$ is already answered. For instance, in case of the existence of tree-shaped models, these do always exist for $\mathcal{ALC}$ concepts defined w.r.t. a TBox, because $\mathcal{ALC}$ has the tree model property, i.e. any $\mathcal{ALC}$ concept satisfiable w.r.t. a TBox also has a model that is tree-shaped.

We focus in this paper on the successfulness problem for a DL and a transformation pair, i.e. we want to obtain general results of the form: given a transformation pair T, for all source pairs S of a given DL $\mathcal{L}_\mathcal{K}$, $\mathsf{successful}(S, T)$ holds. This is the case if τ is a model preserving relation w.r.t. a logic $\mathcal{L}_\mathcal{K}$. For instance, by the virtue of invariance results it holds that, if source and target interpretation of τ are proven to be globally ($\mathcal{ALC}$)-bisimilar, every ($\mathcal{ALC}$) knowledge base satisfied by the source interpretation is also satisfied by the target one.

Ultimately, it would be desirable to not only decide successfulness, but to compute the actual ρ-model for a source pair. Given a particular ρ, it is not obvious how to obtain a suitable definition scheme. For instance, if ρ requires the model to have no isolated vertices, the scheme could either connect or delete those vertices from the input model. So it is not trivial to derive the formulae for the definition scheme merely from the property ρ for the target model. We are interested in the investigation of the relation of ρ to the formulae of a definition scheme for future work.

4 A Transduction for Model Unraveling

We introduce a family of transductions $(\tau_{\ell\text{-}\mathsf{Tree}})_{\ell\in\mathbb{N}}$ as transformation formalism for source pairs $S = (\mathcal{I}, \mathcal{K})$, where $\mathcal{K}$ is an $\mathcal{ALC}$ knowledge base (with an empty ABox). The goal is to achieve a tree(-like) model which is defined as follows.

Definition 6 (Property ℓ-tree-like). *A pointed interpretation $(\mathcal{I}, d)$ satisfies the property* ℓ-tree-like *iff it is one connected component and a partitioning of the edges into 3 sets s.t.*

- *d has no predecessor w.r.t. the first partition of the edges and for every element that is reachable from d by a non-empty path of length $< \ell$ by edges of that partition there is only one predecessor,*
- *each element e that is reachable from d by a path of length ℓ is a node in a directed graph G_e, whose edges belong exclusively to the second partition and each directed graph G_e has at most one node reachable from d by a path of length ℓ*
- *edges in the third partition exclusively have their first component in the directed graph G_e and their second component in the path from d to e.*

Intuitively, we want to unravel arbitrary models for $\mathcal{ALC}$ KBs into a tree of (user definable) depth ℓ. The DL $\mathcal{ALC}$ admits the tree model property as well as the

finite model property, but need not have finite tree models. Hence, some leaves of the tree might need to re-use elements as successors, since the transformation must be model preserving. The idea is to introduce back-loops to the branches of the tree to the most recent suitable ancestor. The resulting model is ℓ-tree-like. As an intuitive example consider the TBox $\{A \sqsubseteq \exists r.A\}$ and the interpretation $\mathcal{I} = (\{a\}, \{(a,a) \in r\})$. Our transformation would create an ℓ long chain of copies of a connected by an r-edge and needs to loop back to some element.

The idea for the unraveling transduction is the following. The "root" element d is copied into the domain of the target interpretation $\mathcal{I}'$ and is the root of the tree-like interpretation. As in the standard tree-unraveling defined for interpretations (see [2]), each element on a path up to length ℓ in $\mathcal{I}$ from the given unraveling element d, is copied into the domain of the target interpretation $\mathcal{I}'$ as often as it is occurring on such a path. These nodes are the "inner nodes" of the tree-like structure. Elements that occur in $\mathcal{I}$ at a greater distance to d than ℓ, are copied only once into the target domain. These are located at the "end" of the tree-structure and are called (complex) leaves—despite the fact that they can have an arbitrary relational neighborhood. There are three kinds of edges: those edges called *branches* that connect the nodes being the root or inner nodes in the tree-structure, those edges called *thickets* that belong to the arbitrary structure within a complex leaf and those edges that point from a node within a complex leaf back to the tree-structure called *back-loops*. If an element of a complex leaf has an r-successor in $\mathcal{I}$, that has already a copy on this branch, the complex leaf is connected to, the transduction introduces a back-loop back into this copy. More precisely, the back-loop is to the youngest ancestor copy on the branch. This design choice creates short back-loops. The underlying idea of the transduction is to identify each domain element by the shortest path from the unraveling node over $\Delta^{\mathcal{I}}$.

We use the following auxiliary predicate to express reachability (confer [7]) defined by the MSO formula:

$$reach(x,y) := \forall X \Big(x \in X \wedge \big(\forall u,v : u \in X \wedge \big(\bigvee_{r \in \Sigma_r} r(u,v)\big) \Rightarrow v \in X\big) \Rightarrow y \in X \Big) \quad (1)$$

in the definition schema $\mathsf{D}_{\ell\text{-}\mathsf{Tree}}$ inducing the unraveling transduction.

Definition 7 (Unraveling transduction). *Let $\mathcal{I}$ be an interpretation, $d \in \Delta^{\mathcal{I}}$ called the* unraveling node*, $\ell \in \mathbb{N}$. Also, let the set of path words of up to length ℓ in $\mathcal{I}$ be $\alpha(\mathcal{I}, \ell) := \{w = x_1 \cdots x_i \in (\Delta^{\mathcal{I}})^i \mid 0 \leq i \leq \ell\}$ and $w = x_1 \cdots x_n$ and $w' = y_1 \cdots y_m$ be two words. The definition scheme (with $\mathsf{Par} = \{y\}$ and $\mathsf{AuxVar} = \{u, v\}$) is*

$$\mathsf{D}_{\ell\text{-}\mathsf{Tree}} = \big(\chi,\ (\delta_w)_{w \in \alpha(\mathcal{I},\ell)},\ (\theta_{C,w})_{C \in \Sigma_{\mathsf{A}}, w \in \alpha(\mathcal{I},\ell)},\ (\eta_{r,(w,w')})_{r \in \Sigma_{\mathsf{r}}, (w,w') \in \alpha(\mathcal{I},\ell)^2}\big),$$

where the components are defined as follows:

– *Precondition:* $$\chi(y) := \forall x : x \neq y \rightarrow reach(x,y) \quad (2)$$

– *Domain formulae are defined by means of the following formulae:*

$$\delta_w^{\mathsf{root}}(y,u) := u = y \tag{3}$$

$$\delta_w^{\mathsf{inner}}(y,u) := \bigwedge_{i=1}^{|w|} \Big(\bigvee_{\mathsf{r}\in\Sigma_\mathsf{r}} r(x_i,x_{i+1}) \Big) \wedge (x_1 = y) \wedge (x_{|w|+1} = u) \tag{4}$$

$$\delta_w^{\mathsf{leaf}}(y,u) := \delta_{w'}(y,x_{|w|}) \wedge reach(x_{|w|},u) \wedge \Big(\bigwedge_{i=1}^{|w|} u \neq x_i \Big) \tag{5}$$

$$\delta_w(y,u) := \begin{cases} \delta_w^{\mathsf{root}}(y,u) & , \textit{ if } |w| = 0 \\ \delta_w^{\mathsf{inner}}(y,u) & , \textit{ if } |w| \leq \ell - 1 \\ \delta_w^{\mathsf{leaf}}(y,u) & , \textit{ if } |w| = \ell \ \&\ w = w' \cdot x_\ell \end{cases} \tag{6}$$

The variable x_i used in the formulae of $\mathsf{D}_{\ell\text{-}\mathsf{Tree}}$ needs to be mapped by λ to the according $x_i \in w$.[1]

– *Concept name formulae:* $$\theta_{\mathsf{A},w}(u) := \mathsf{A}(u) \tag{7}$$

– *Role name formulae: $\eta_{\mathsf{r},(w,w')}$ are defined by means of the following formulae:*

$$\eta_{r,(w,w')}^{\mathsf{thicket}}(u,v) := r(u,v) \tag{8}$$

$$\eta_{r,(w,w')}^{\mathsf{branch}}(u,v) := r(u,v) \wedge w' = w \cdot u \tag{9}$$

$$\eta_{r,(w,w')}^{\mathsf{back\text{-}loop}}(u,v) := r(u,v) \wedge \Big(\bigwedge_{i=1}^{m} x_i = y_i \Big) \wedge \Big(\bigwedge_{i=m+1}^{n} x_i \neq v \Big) \tag{10}$$

$$\eta_{r,(w,w')} := \begin{cases} \eta_{r,(w,w')}^{\mathsf{thicket}}(u,v) & \textit{if } |w| = |w'| = \ell \\ \eta_{r,(w,w')}^{\mathsf{branch}}(u,v) & \textit{if } |w'| = |w| + 1 \ \&\ |w'| < \ell \\ \eta_{r,(w,w')}^{\mathsf{back\text{-}loop}}(u,v) & \textit{if } |w'| < |w| \ \&\ |w| = \ell \end{cases} \tag{11}$$

The definition schema $\mathsf{D}_{\ell\text{-}\mathsf{Tree}}$ together with the assignment $\lambda_{\mathsf{Par}} : y \mapsto d$, then defines the transduction relation τ_ℓ. Other (undefined) cases are set to the Boolean constant $\bot$.

Note that the index set is not the natural numbers, but the set of words over $\Delta^\mathcal{I}$ up to length ℓ. This is w.l.o.g. as there always is an order on that index set isomorphic to the natural numbers, e.g. the lexicographic order. According to Definition 2, every transduction induced by a definition scheme can only increase the domain size linearly. Using $\alpha(\mathcal{I},\ell)$ as index set, however, the increase of the domain size is limited to $|\Delta^\mathcal{I}| \cdot |\alpha(\mathcal{I},\ell)|$, and $|\alpha(\mathcal{I},\ell)| = \sum_{k=1}^{\ell} |\Delta^\mathcal{I}|^k$, which is not linear but still polynomial.

[1] This is an element in $\Delta^\mathcal{I}$ by definition, i.e. the transduction uses more parameters than just the unraveling node d. For eased readability we use x_i for both the variable in the formulae and the domain elements of $\mathcal{I}$ since we identify them with λ.

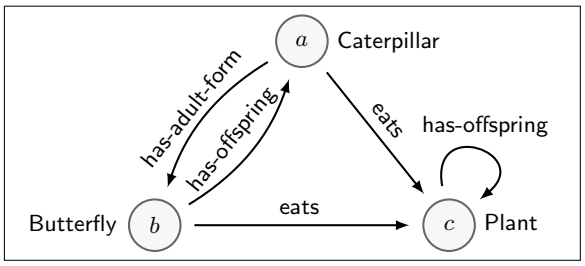

Fig. 1. Model $\mathcal{I}_{ex}$ of TBox $\mathcal{T}_{ex}$.

We before we investigate formal properties of the unravelling transduction, we illustrate the unraveling of a model of $\mathcal{T}_K$ by an example in the realm of biology. Consider the following TBox that speaks about caterpillars, butterflies and plants. It states that all butterflies eat plants and have caterpillars as offspring. Also, every caterpillar eats plants and has a butterfly as an adult form. Lastly, every plant has a plant as offspring.

$$\begin{aligned}\mathcal{T}_{ex} := \{\ & \mathsf{Butterfly} \sqsubseteq (\exists \mathsf{has\text{-}offspring}.\mathsf{Caterpillar}) \sqcap (\exists \mathsf{eats}.\mathsf{Plant}),\\ & \mathsf{Caterpillar} \sqsubseteq (\exists \mathsf{has\text{-}adult\text{-}form}.\mathsf{Butterfly}) \sqcap (\exists \mathsf{eats}.\mathsf{Plant}),\\ & \mathsf{Plant} \sqsubseteq \exists\, \mathsf{has\text{-}offspring}.\mathsf{Plant}\ \}\end{aligned}$$

Figure 1 depicts a model $\mathcal{I}_{ex}$ for the TBox $\mathcal{T}_{ex}$ using only three domain elements. Concept names are written next to the nodes and the labeled arrows depict roles. Such a succinct model could be the outcome of the reasoning process of reasoners that aim at keeping the domain of the model small by eagerly reusing domain elements and blocking the introduction of new elements possible—a strategy known as anywhere blocking that is implemented by the HermiT reasoner [8].

Model $\mathcal{I}_{ex}$, however, has some shortcomings when it comes to explaining the concepts under consideration. For instance, the offspring of a butterfly is certainly not the caterpillar that this butterfly developed from. Also, a regular plant not its own offspring. A model unraveling might help to resolve some of these artifacts due to reasoner optimization.

Figure 2 shows the 2-unraveling of $\mathcal{I}_K$. The parts of the unraveled model in solid lines highlight the tree of depth 2 and the dotted lines and are being added for the transformation to be model preserving. In the part of the model drawn in solid lines, we now have that the offspring of a butterfly is no longer the caterpillar it emerged from and also, plants have other (freshly introduced) plants as offspring, too. For explanation purposes, users might only be interested in the non-dotted part and an implementation could simply omit the parts drawn in dotted line if the user permits.

We show now that the unraveling transduction is successful for all $\mathcal{ALC}$ KBs. To this end we employ bisimulations between source and target interpretations.

Lemma 1. *The relation* $\simeq := \{(x, (x', w)) \mid (x', w) \in \Delta^{\tau_{\ell\text{-}\mathsf{Tree}}(\mathcal{I})}\ \&\ x = x'\})$ *is a bisimulation between* $\mathcal{I}$ *and* $\tau_{\ell\text{-}\mathsf{Tree}}(\mathcal{I})$.

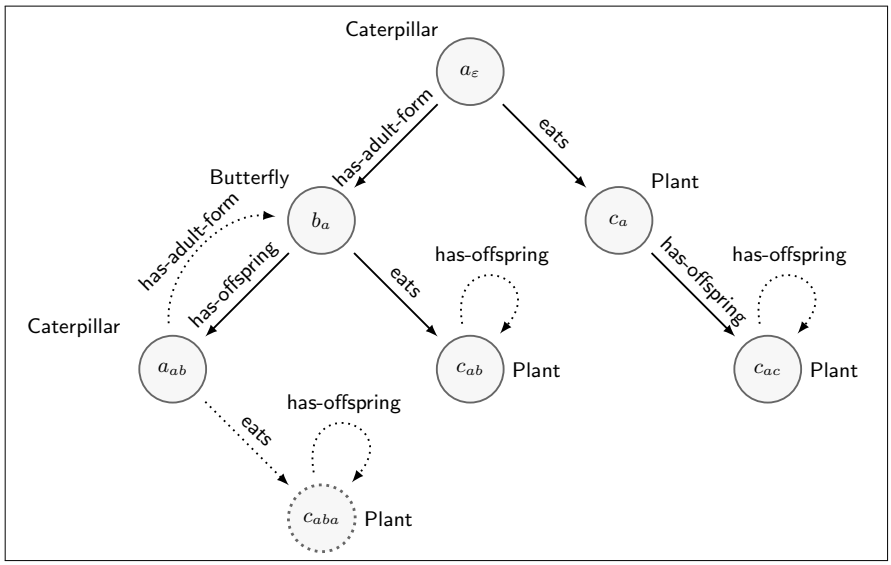

Fig. 2. The 2-unraveling of model $\mathcal{I}_K$ into a 2-tree-like structure.

Proof. We denote $\tau_{\ell\text{-Tree}}(\mathcal{I})$ *by* $\mathcal{I}'$*. Since* x' *in* $(x', w) \in \Delta^{\mathcal{I}'}$ *is an element of* $\Delta^{\mathcal{I}}$*, we can state* $x = x'$ *as condition. Assume* $\mathcal{I} \models \chi$*, so that* $\tau_{\ell\text{-Tree}}(\mathcal{I})$ *is defined. We need to show three conditions for this claim.*

First, $x \simeq (x', w)$ *implies* $x \in A^{\mathcal{I}} \Leftrightarrow (x', w) \in A^{\mathcal{I}'}$ *for all* $x \in \Delta^{\mathcal{I}}$*,* $(x', w) \in \Delta^{\mathcal{I}'}$ *and* $A \in \Sigma_{\mathsf{A}}$*; this condition follows directly from concept name formula (7).*

Second, $x \simeq (x', w)$ *and* $(x, y) \in r^{\mathcal{I}}$ *implies the existence of* (y', w') *such that* $((x', w), (y', w')) \in r^{\mathcal{I}'}$ *and* $y \simeq (y', w)$ *for all* $x, y \in \Delta^{\mathcal{I}}$*,* $(x', w), (y', w') \in \Delta^{\mathcal{I}'}$ *and all* $r \in \Sigma_{\mathsf{r}}$*. Assume* $(x, y) \in r^{\mathcal{I}}$ *and* $x \simeq (x', w)$*. Since* $\mathcal{I}$ *satisfies the precondition (2), there is a path in* $\mathcal{I}$ *from the unraveling node* d *to* x *and hence to* y*. Now, domain formulae (3) and (4) create a fresh copy for each element ending in a* d*-path in* $\mathcal{I}$ *up to length* ℓ*. If the path from* d *to* x *is of length at most* ℓ *in* $\mathcal{I}$*, we have to distinguish two cases, because an* r*-successor could have been copied before in this path. If the* r*-successor has been copied before, domain formula (4) is true for* y *and is hence copied into the domain of* $\mathcal{I}'$*, i.e. there is an* $(y, w) \in \Delta^{\mathcal{I}'}$*. If the* r*-successor* y *of* x *in* $\mathcal{I}$ *has not been copied yet (because the* d*-path is longer than* ℓ*), domain formula (5) is true for* y *and hence* y *is copied into the domain of* $\mathcal{I}'$ *as well. To see that the role relationships are set correctly, we consider two cases. The considered path from* d *to* y *over* x *in* $\mathcal{I}$ *is of length up to* ℓ *or it is not. Role formula (9) connects two nodes* (x, w) *and* (y, w') *with an* r*edge if* $r(x, y)$ *holds in* $\mathcal{I}$ *and* $w = w \cdot x$*. In the latter case, we again distinguish two cases. If the* r*-successor* y *of* x *in* $\mathcal{I}$ *is already reachable over a prefix of path of* (x, w)*, then (10) adds the* r*edge to an earlier copy of* y *in* $\mathcal{I}'$*. If there is no copy of a prefix of* w*, relation formula (8) connects* (x', w') *and* (y', w') *whenever* x *is connected to* y *in* $\mathcal{I}$*.*

Third, $x \simeq (x, w)$ *and* $((x', w), (y', w')) \in r^{\mathcal{I}'}$ *implies the existence of* $y \in \Delta^{\mathcal{I}}$ *such that:* $(x, y) \in r^{\mathcal{I}}$ *and* $y \simeq (y', w')$ *for all* $x \in \Delta^{\mathcal{I}}$, $x', y' \in \Delta^{\mathcal{I}'}$ *and* $r \in \Sigma_{\mathsf{r}}$. *Similarly to the previous argument, assume we are given* $(x \simeq (x', w)$ *and* $(x', y') \in r^{\mathcal{I}'}$. *Then, there is a path in* $\mathcal{I}'$ *from* d *over* x *to* y, *since* y' *is copied if and only if there is such a path according to domain formulae (4) and (5). And here as well as in the previous condition, an* r*-edge between* (x, w) *and* (y, w) *is added if and only if there is an* r*-edge between* x *and* y *in* $\mathcal{I}$.

Lemma 2. *Let the set* $\mathcal{C} := \{[(x, w)]_{\equiv}\}$ *of equivalence classes be defined by the equivalence relation* $(x, w) \equiv (x', w')$ *iff* $x = x'$ *over* $\Delta^{\tau_{\ell\text{-}\mathsf{Tree}}(\mathcal{I})}$. *Then* $\mathcal{C}$ *is a partitioning of the domain of* $\tau_{\ell\text{-}\mathsf{Tree}}(\mathcal{I})$.

Corollary 1. *Let* $\mathcal{I}$ *be an interpretation and* $\tau_{\ell\text{-}\mathsf{Tree}}$ *the unraveling transduction for some* $\ell \in \mathbb{N}$. *Then, if* $\tau_{\ell\text{-}\mathsf{Tree}}(\mathcal{I})$ *is defined,* $\mathcal{I}$ *and* $\tau_{\ell\text{-}\mathsf{Tree}}(\mathcal{I})$ *are globally bisimilar.*

Proof. Recall the relation $\simeq$ *from Lemma 1. We show that for each* $e \in \Delta^{\mathcal{I}}$, *there is an element* $e' \in \Delta^{\tau_{\ell\text{-}\mathsf{Tree}}(\mathcal{I})}$, *such that* $(\mathcal{I}, e) \simeq (\tau_{\ell\text{-}\mathsf{Tree}}(\mathcal{I}), e')$, *and vice versa. To achieve this, we use the partitioning* $\mathcal{C}$ *from Lemma 2. For each element in the domain of* $\mathcal{I}$ *exists exactly one corresponding element in* $\mathcal{C}$. *Following the argument in Lemma 1 and since* $\tau_{\ell\text{-}\mathsf{Tree}}(\mathcal{I})$ *is defined, for each reachable element from the unraveling node, there is at least one copy. Hence, the relation* $\simeq$ *is a bijection between* $\mathcal{I}$ *and* $\mathcal{I}'$ *and hence, we have that* $\mathcal{C}$ *is a partitioning. We also have that for each representative* (e, w) *of* $[(e, w)]_{\equiv}$, *it is the case that* $e \simeq (e, w)$ *as defined and hence we have that* $\simeq$ *is global. So, we have that for each elements of the domains of* $\mathcal{I}$ *and* $\mathcal{I}'$, *there is at least one bisimilar element in the other one respectively.*

Theorem 1. *For all source pairs* $S = (\mathcal{I}, \mathcal{T})$, *where* $\mathcal{T}$ *is an* $\mathcal{ALC}$*-TBox and* $\mathcal{I}$ *an interpretation holds: if* $\tau_{\ell\text{-}\mathsf{Tree}}(\mathcal{I})$ *is defined, then* $\mathsf{successful}(S, T)$.

Proof. Direct consequence of Corollary 1 and Definition 6.

Note that the computational complexity of applying $\tau_{\ell\text{-}\mathsf{Tree}}$ to an input interpretation is the sum of evaluating the definition scheme over this interpretation. The amount of formulae to evaluate for $\tau_{\ell\text{-}\mathsf{Tree}}$ is polynomial in size of $|\Delta^{\mathcal{I}}|$ as mentioned before. To compute reachability (here expressed by the predicate $reach(x, y)$), well-known PTIME time algorithms exist.

Considerations on model transformation for $\mathcal{EL}$. Since the DL $\mathcal{EL}$ does not use negation, every $\mathcal{EL}$-KB has a model. A GCI formulated in $\mathcal{EL}$ is satisfied in one interpretation $\mathcal{I}$ is also satisfied in another interpretation $\mathcal{I}$' iff there is a homomorphism $h : \mathcal{I} \mapsto \mathcal{I}'$. This seems clear from the fact that homomorphisms are relation preserving mappings. Hence, if we instantiate our framework such that $\mathcal{L}_{\mathcal{K}} = \mathcal{EL}$, we need to show that τ is a homomorphism in order to guarantee successfulness for each $\mathcal{EL}$ TBox. As global bisimulation implies the existence of a homomorphism, the $\tau_{\ell\text{-}\mathsf{Tree}}$ transduction preserves models of $\mathcal{EL}$

knowledge bases. An important difference to $\mathcal{ALC}$ is that $\mathcal{EL}$ has the canonical model property, which means that there always exists a model that can homomorphically be embedded into any other model of the same $\mathcal{EL}$-knowledge base. Canonical models are the basis for reasoning in $\mathcal{EL}$ and they are computed by all dedicated $\mathcal{EL}$ reasoners such as ELK [9]. Thus an interesting variant of our framework with $\mathcal{L}_\mathcal{K} = \mathcal{EL}$ is that by the $\mathcal{EL}$-KB $\mathcal{K}$ also the source model $\mathcal{I}$ is determined and a transduction can make much stronger assumptions on its input. Whether this gives better computational properties for some properties ρ needs to be investigated.

5 Conclusions and Future Work

We have defined and discussed a general framework for model transformation in DL—motivated by the task to generate models of DL TBoxes that are suitable for explanation. Based on this framework we have formally defined a collection of reasoning problems. By means of results from model theory one can address some of these reasoning problems. We have defined and investigated an ℓ-tree unraveling of (finite) interpretations as a concrete instantiation of the framework and have shown that the corresponding transduction is model preserving with respect to any TBox formulated in the logic $\mathcal{ALC}$.

In this paper we have mainly discussed the decision problems related to the existence of ρ-models. For the application of generating models that facilitate explanations, the corresponding computation problems are clearly of bigger interest as they pave the way to automatic support. However, it is not obvious how to use a property ρ to derive a suitable definition scheme for that property. In practice it would probably suffice to offer a fixed catalog of properties to a user in order to display a model with one or even several of the properties from the catalog. Having a catalog of properties for which there are definition schemes and results like Theorem 1 (with respect to a given logic), the transformations could be implemented in a tool, in which the user specifies the source and transformations pairs. Populating such a catalog with suitable model properties should be done in regard of results from the fields of cognitive sciences or visualization.

There are many extensions of this initial work to be studied in future work. An obvious extension of the work presented here, is to use ABoxes as well when computing or transforming models. As ABoxes add named individuals to models, transformations have to obey the unique name assumption, which says that differently named objects have to be mapped to different domain elements in interpretations. This assumption must not be violated by transformations.

The framework is not tied to transductions and as an alternative transformation formalism, graph rewriting systems are a interesting option. A comparison of graph rewriting systems and graph transductions in regard of treating models derived from DL knowledge bases is future work, as well.

Acknowledgments. We would like to thank Heiko Vogler and the anonymous reviewers for their detailed and helpful remarks.

References

1. do Amaral, F.N., Martins, C.B.: Visualization of description logic models. In: Proceedings of the 21st International Workshop on Description Logics (DL 2008). CEUR Workshop Proceedings, vol. 353 (2008)
2. Baader, F., Horrocks, I., Lutz, C., Sattler, U.: An Introduction to Description Logic. Cambridge University Press (2017)
3. Bauer, J., Sattler, U., Parsia, B.: Explaining by example: Model exploration for ontology comprehension. In: Proceedings of the 22nd Int. Workshop on Description Logics (DL 2009). CEUR Workshop Proceedings, vol. 477 (2009)
4. Baumgartner, P., Fuchs, A., De Nivelle, H., Tinelli, C.: Computing finite models by reduction to function-free clause logic. J. of Applied Logic **7**(1), 58–74 (2009)
5. van Benthem, J.: Modal Correspondence Theory. Ph.D. thesis, Universiteit van Amsterdam (1976)
6. Brandt, S.: Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and—what else? In: Proceedings of the 16th European Conference on Artificial Intelligence (ECAI-2004). pp. 298–302. IOS Press (2004)
7. Courcelle, B., Engelfriet, J.: Graph Structure and Monadic Second-Order Logic: A Language-Theoretic Approach. Cambridge University Press (2012)
8. Glimm, B., Horrocks, I., Motik, B., Stoilos, G., Wang, Z.: HermiT: an OWL 2 reasoner. Journal of Automated Reasoning **53**(3), 245–269 (2014)
9. Kazakov, Y., Krötzsch, M., Simančík, F.: The incredible ELK. Journal of Automated Reasoning **53**(1), 1–61 (2014)
10. Peñaloza, R.: Explaining axiom pinpointing. In: Description Logic, Theory Combination, and All That - Essays Dedicated to Franz Baader on the Occasion of His 60th Birthday. LNCS, vol. 11560, pp. 475–496. Springer (2019)
11. Rozenberg, G.: Handbook of graph grammars and computing by graph transformation, vol. 1. World scientific (1997)
12. Schild, K.: A correspondence theory for terminological logics: Preliminary report. In: Mylopoulos, J., Reiter, R. (eds.) Proc. IJCAI'91. pp. 466–471 (1991)
13. Tobies, S.: Complexity Results and Practical Algorithms for Logics in Knowledge Representation. Doctoral thesis, Rheinisch-Westfälische Technische Hochschule Aachen, Aachen, Germany (2001)
14. Tsarkov, D., Horrocks, I.: FaCT++ description logic reasoner: System description. Journal of Automated reasoning pp. 292–297 (2006)
15. Vogler, H., Engelfriet, J.: A Büchi-Elgot-Trakhtenbrot theorem for automata with MSO graph storage. Discrete Mathematics & Theoretical Computer Science **22** (2020)