



TECHNISCHE
UNIVERSITÄT
DRESDEN

Technische Universität Dresden
Institute for Theoretical Computer Science
Chair for Automata Theory

LTCS–Report

Optimal Repairs in the Description Logic \mathcal{EL} Revisited (Extended Version)

Franz Baader Patrick Koopmann Francesco Kriegel

LTCS-Report 23-03




This is an extended version of an article accepted at the
18th European Conference on Logics in Artificial Intelligence
(JELIA 2023).

Postal Address:
Lehrstuhl für Automatentheorie
Institut für Theoretische Informatik
TU Dresden
01062 Dresden

<http://lat.inf.tu-dresden.de>

Visiting Address:
Nöthnitzer Str. 46
Dresden

Optimal Repairs in the Description Logic \mathcal{EL} Revisited (Extended Version)

Franz Baader^{1,2} , Patrick Koopmann¹ , and Francesco Kriegel¹ 

¹ Theoretical Computer Science, Technische Universität Dresden, Dresden, Germany
`firstname.lastname@tu-dresden.de`

² Center for Scalable Data Analytics and Artificial Intelligence (ScaDS.AI)
Dresden/Leipzig, Germany

Abstract. Ontologies based on Description Logics may contain errors, which are usually detected when reasoning produces consequences that follow from the ontology, but do not hold in the modelled application domain. In previous work, we have introduced repair approaches for \mathcal{EL} ontologies that are optimal in the sense that they preserve a maximal amount of consequences. In this paper, we will, on the one hand, review these approaches, but with an emphasis on motivation rather than on technical details. On the other hand, we will describe new results that address the problems that optimal repairs may become very large or need not even exist unless strong restrictions on the terminological part of the ontology apply. We will show how one can deal with these problems by introducing concise representations of optimal repairs.

1 Introduction

Description Logics (DLs) [6, 7] are a prominent family of logic-based knowledge representation formalisms, which offer a good compromise between expressiveness and the complexity of reasoning and are the formal basis for the Web ontology language OWL.³ In a DL ontology, the important notions of the application domain are introduced as background knowledge in the *terminology* (*TBox*), and then these notions are used to represent a specific application situation in the *ABox*. The DLs of the \mathcal{EL} family have drawn considerable attention since their reasoning problems are tractable [4], but they are nevertheless expressive enough to represent ontologies in many application domains, such as biology and medicine.⁴ For instance, the medical ontology SNOMED CT employs \mathcal{EL} and contains the following *concept inclusion* (*CI*) in its TBox:

$$\begin{aligned} \text{Common_cold} \sqsubseteq & \text{Disease} \sqcap \exists \text{causative_agent. Virus} \\ & \sqcap \exists \text{finding_site. Upper_respiratory_tract_structure} \\ & \sqcap \exists \text{pathological_process. Infectious_process,} \end{aligned}$$

³ <https://www.w3.org/TR/owl2-overview/>

⁴ see. e.g., <https://bioportal.bioontology.org> and <https://www.snomed.org/>

which says that a common cold is a disease that is caused by a virus, can be found in the upper respiratory tract, and has as pathological process an infectious process. A GP can then employ this concept to store in the ABox that patient Alice is diagnosed with common cold using the *concept assertion* $(\exists \text{has_diagnosis. Common_cold})(\text{alice})$. The GP’s ABox may also contain the information that Charles is Alice’s father, expressed as *role assertion* $\text{has_father}(\text{alice}, \text{charles})$, which might be of interest in the context of hereditary diseases.

Like all large human-made digital artefacts, the ontologies employed in such applications may contain errors, and this problem gets even worse if parts of the ontology (usually the ABox) are automatically generated by inexact methods based on information retrieval or machine learning. Errors in ontologies are often detected when the reasoner generates a consequence that formally follows from the knowledge base, but is incorrect in the sense that it does not hold in the application domain that is supposed to be modelled. For example, in a previous version of SNOMED CT, the concept “Amputation of finger” was classified as a subconcept of “Amputation of hand,” which is fortunately wrong in the real world. To correct such errors in large ontologies, the *knowledge engineer (KE)* should be supported by an appropriate *repair tool*. Such a tool receives as input one or more consequences of the given ontology that are unwanted, and it should return one or more repaired ontologies that no longer have these consequences (called *repairs*). The KE can then choose one of the computed repairs and either use it as the new ontology, or continue the repair process from it if other unwanted consequences are detected. Of course, it makes no sense to use as a repair an arbitrary ontology that does not have the unwanted consequences. The repaired ontology should (a) not introduce new information and (b) be as close as possible to the original ontology. There are different possibilities for how to formalize these conditions.

The *classical approaches* for ontology repair return maximal subsets of the ontology that do not have the unwanted consequence, and employ methods inspired by model-based diagnosis [45] to compute these sets [20, 44, 46]. Thus, these approaches interpret the above conditions in a syntactic way: (a) is read as “no new axioms” and (b) is realized by the maximality condition. In [18] we called classical repairs that satisfy this maximality condition *optimal classical repairs*. While these approaches preserve as many of the axioms in the ontology as possible, they need not preserve a maximal amount of consequences, and they are syntax-dependent. For example, consider the ABoxes $\mathcal{A} := \{(A \sqcap B)(a)\}$ and $\mathcal{B} := \{A(a), B(a)\}$, which both say that individual a belongs to the concepts A and B , and are thus equivalent. However, with respect to the unwanted consequence $A(a)$, the ABox \mathcal{A} has the empty ABox as only optimal classical repair, whereas \mathcal{B} has the optimal classical repair $\{B(a)\}$. Thus, the latter repair retains the consequence $B(a)$, whereas the former does not. To overcome this problem, more gentle repair approaches have been introduced, e.g., in [18, 28, 31, 36, 47]. The basic idea underlying these approaches is to replace some axioms of the ontology by weaker ones, rather than just removing them, as in the classical

approach. In our example, one can replace the axiom $(A \sqcap B)(a)$ in the ABox \mathcal{A} with the weaker axiom $B(a)$, and thus retain the consequence $B(a)$ even if one starts with \mathcal{A} rather than \mathcal{B} . However, these *gentle repairs* are still dependent on the syntactic structure of the axioms in the ontology, and how well they realize condition (b) depends on the employed weakening relation between axioms and the strategy used to apply it.

Providing the KE with syntax-dependent repair tools is not in line with the *functional approach* to knowledge representation [24,37] adopted by DLs. In this approach, the syntactic structure of the axioms in the ontology is supposed to be irrelevant. What counts is what queries are entailed by the ontology, which in DLs are usually *instance queries (IQ)* or *conjunctive queries (CQ)*. In this functional setting, (a) should be read as “no new consequences” (expressed in the adopted query formalism) and (b) as preserving a maximal set of such consequences. This leads us to the definition of an *optimal repair* [9,18], which is an ontology that does not have the unwanted consequences, is entailed by the original ontology (thus realizing property (a)), and preserves a maximal amount of consequences in the sense that there is no repair (i.e., no ontology satisfying the first two properties) that strictly entails it (property (b)). Entailment can be *IQ-entailment* or *CQ-entailment*, depending on whether we are interested only in instance queries, or also in conjunctive queries [39]. Maximizing the retained consequences is also motivated by the following observation. All the repair tool knows is the original ontology and the consequences that should be removed, which are specified in what we call a *repair request*. If it were to remove more consequences than are strictly needed to satisfy the repair request, then the decision which additional consequences to remove would be a random choice by the tool, not based on any application knowledge, which is held by the KE. In case the optimal repair retains consequences that should be removed, the KE needs to specify this in a subsequent repair request.

If a *repair problem* consisting of an ontology and a repair request does not have a repair, then it cannot have an optimal one. In general, however, optimal repairs of repair problems that have a repair need not exist either, even in the simple setting of \mathcal{EL} ABoxes without a TBox. This is illustrated in the following example, where the ABox $\mathcal{A} = \{V(n), \ell(n, n)\}$ says that Narcissus is a vain individual that loves itself, and the repair request $\mathcal{R} = \{V(n)\}$ wants us to remove the consequence that Narcissus is vain. Intuitively, to obtain a repair, we must remove $V(n)$. However, since all assertions of the form $\exists \ell.(V \sqcap (\exists \ell.)^k \top)(n)$, saying that Narcissus loves a vain individual that is the starting point of a loves-chain of length k , are consequences of \mathcal{A} and can be added to $\{\ell(n, n)\}$ without entailing $V(n)$, it is easy to see that there is no finite \mathcal{EL} ABox that is an optimal repair. In fact, since Narcissus is no longer vain, the retained cycle $\ell(n, n)$ cannot be used to generate the loves-chains of arbitrary length starting from a vain individual. Even if a given repair problem has optimal repairs, they may not *cover all repairs* in the sense that every repair is entailed by an optimal one. To see this, we can look at a modified version of the above example. Consider the ABox $\mathcal{B} = \{k(t, n), V(n), \ell(n, n)\}$, which contains the

additional information that Tiresias knows Narcissus, and the repair request $\mathcal{Q} = \{(\exists k.V)(t)\}$. Removing $k(t, n)$ from \mathcal{B} yields an optimal repair. However, there are also repairs that retain this assertion, but there is no optimal one among them for the same reason as in the previous example. Thus, if the KE is only offered the optimal repair $\{V(n), \ell(n, n)\}$ by the repair tool, the repair options that retain the assertion $k(t, n)$ are missed. This illustrates that the use of optimal repairs in a repair tool requires a setting where the optimal repairs always cover all repairs.

This can be achieved by using a more general notion of ABoxes, called *quantified ABoxes* (*qABoxes*) [19], where in addition to the usual named individuals we also have anonymous objects, which are represented as (existentially quantified) variables. In our Narcissus example, an optimal repair of \mathcal{A} for \mathcal{R} is obtained by removing $V(n)$ and introducing an anonymous vain and self-loving lover of Narcissus, which yields the qABox $\exists \{x\}. \{\ell(n, n), \ell(n, x), \ell(x, n), \ell(x, x), V(x)\}$. Note that we could not have used a named individual b instead of the variable x since then the resulting ABox would have entailed instance relationships for b , such as $V(b)$, that are not entailed by \mathcal{A} . One might think that retaining a consequence like $(\exists \ell.V)(n)$ is not justified since one of the reasons for this being a consequence of \mathcal{A} , namely $V(n)$, has been removed. However, with this argument, we would be back at the classical repair approach. As argued above, since the repair request only specifies that $V(n)$ should no longer be a consequence, other consequences like $(\exists \ell.V)(n)$ should not be lost unless this is needed to remove $V(n)$.

In [19] we consider a setting where ontologies are qABoxes and the repair requests consist of entailed $\mathcal{E}\mathcal{L}$ instance relationships.⁵ Given such a repair problem, we show how to construct a finite set of repairs, called the *canonical repairs*, which cover all repairs. The canonical repairs are of exponential size, and there may be exponentially many of them. Not every canonical repair is optimal, but due to the covering property, the set of them contains all optimal repairs up to equivalence. The set of optimal repairs can thus be obtained by removing non-optimal canonical repairs, i.e., ones that are strictly entailed by another canonical repair, and this set covers all repairs. The construction of the canonical repairs is actually the same for the CQ and the IQ case. The only difference is that, when removing non-optimal canonical repairs, the respective entailment relation must be used. Since CQ-entailment implies IQ-entailment, but not vice versa, more canonical repairs may be removed as non-optimal in the IQ setting. In addition, since CQ-entailment is NP-complete and IQ-entailment is tractable, the complexity of removing non-optimal repairs is higher in the CQ case.

The differences between the CQ and the IQ case get more pronounced if we add an $\mathcal{E}\mathcal{L}$ TBox. In [9], we assume that this *TBox is correct*, and thus should not be changed in the repair process. In order to adapt the approach and the results of [19] to this setting, the first step is to *saturate* the given qABox w.r.t.

⁵ The paper [19] actually calls repairs “compliant anonymisations” and repair requests “privacy policies” since it considers a situation where consequences are to be removed not because they are incorrect, but since this information should be hidden.

the TBox, to reduce entailment with TBox to entailment without TBox. For the IQ case, such a saturation always exists and can be computed in polynomial time. For the CQ case, a finite saturation need not exist in general. However, for *cycle-restricted TBoxes* [3], it always exists, but may be of exponential size. Continuing the repair process with the saturated qABox, we still need to take the TBox into account when defining canonical repairs, to ensure that consequences that have been removed from the qABox cannot be reintroduced by the TBox. With this adapted notion of canonical repairs, we obtain the same results as for the case without TBox. The canonical repairs cover all repairs and can be computed in exponential time. From them the set of all optimal repairs can be obtained by removing non-optimal ones using entailment test [9]. This works both for the IQ and the CQ case, but in the latter only if we can compute a finite saturation, which is always the case if the TBox is cycle-restricted. For TBoxes that are not cycle-restricted, optimal repairs need not exist in the CQ case. For example, with respect to the TBox $\{V \sqsubseteq \exists \ell.V, \exists \ell.V \sqsubseteq V\}$, which says that vain individuals are exactly the ones that love a vain individual, the qABox $\{V(n)\}$ does not have an optimal repair for the repair request $\mathcal{R} = \{V(n)\}$. Intuitively, the reason is that the qABox together with the TBox implies the existence of arbitrarily long loves-chains starting from n , which are no longer entailed by the TBox if $V(n)$ is removed (see Example 9 in [13] for a more detailed argument). One might think that the first GCI $V \sqsubseteq \exists \ell.V$ is enough to destroy existence of an optimal repair. This is, however, not the case. Without the second GCI one can introduce an anonymous vain individual x that is loved by n and loves itself to obtain an optimal repair.

In the *first part* of the paper (Section 2 and Section 3), we will describe the repair approaches developed in our previous work [9, 19], but with an emphasis on motivation rather than on technical details. The *second part* of the paper (Section 4 and Section 5) describes new result. We will consider more *concise representations* of optimal repairs, to deal both with the exponential size of canonical repairs in the IQ case and the non-existence problem w.r.t. cyclic TBoxes in the CQ case.

The former problem is due to the fact that the canonical repairs employed in our approach are by construction of exponential size. To alleviate this problem, we have, on the one hand, developed in [9] an optimized algorithm for computing repairs, which yields *optimized repairs* that are equivalent to the canonical ones, but in most cases considerably smaller, though in the worst case they may still be exponential. On the other hand, each canonical repair is induced by a so-called *repair seed*, whose size is polynomial in the size of the TBox and the repair request. We have seen in [15] that, for the IQ case, one can compute consequences of canonical repairs and check IQ-entailment between them by working only with the seed functions inducing them. This way, the exponential blow-up due to the construction of the canonical repair can be avoided. In Section 4, we report on experimental results that compare the performance on answering instance queries between the optimized repairs and the canonical ones represented by seed functions.

In Section 5, we show that, also in the CQ case, optimal repairs always exist and cover all repairs if we allow for certain infinite, but finitely represented qABoxes. To be more precise, we introduce the notion of a *shell unfolding* of a given qABox, which basically unravels parts of the qABox into (possibly infinite) trees. The shell unfoldings of IQ-saturations turn out to be CQ-saturations, and this also works for cyclic TBoxes. If we then consider the canonical IQ-repairs for a given repair problem, then we can prove that their shell unfoldings yields a set of (possibly infinite) CQ-repairs that cover all CQ-repairs. In addition, consequences from such shell unfolded repairs and entailment between them can be decided based on their finite representation without an increase in complexity. Thus, one can work with them as if they were finite.

This extended version contains all technical details and proofs not included in the conference article [8] for space restrictions. Note that numbering of definitions, lemmas, etc. differs and that the writing style is more technical.

2 Preliminaries

First, we briefly explain the knowledge representation and reasoning capabilities of the description logic \mathcal{EL} . Like other DLs, it allows to assign individuals to concepts, to interrelate individuals by roles, to define compound concepts, and to express inclusion between concepts. The smallest pieces used to represent the knowledge about the domain of interest are collected in the *signature* Σ , which is a set of *individual names*, *concept names*, and *role names*. Every concept name A and the *top concept* \top are \mathcal{EL} *concept descriptions*; if C, D are \mathcal{EL} concept descriptions, then the *conjunction* $C \sqcap D$ is an \mathcal{EL} concept description; for every role name r and every \mathcal{EL} concept description C , the *existential restriction* $\exists r.C$ is an \mathcal{EL} concept description. Repetitions and order in conjunctions are irrelevant, as are nestings of them. Thus, we often use the syntactic sugar $\prod\{C_1, \dots, C_n\} := C_1 \sqcap \dots \sqcap C_n$ and $\prod\emptyset := \top$. An *atom* is either a concept name or an existential restriction. Each \mathcal{EL} concept C is a conjunction of atoms, i.e., $C = \prod \text{Conj}(C)$ for a set $\text{Conj}(C)$ of atoms, which we call the *top-level conjuncts* of C . Assertional knowledge is expressed by *concept assertions* $C(a)$ and *role assertions* $r(a, b)$ composed of a concept description C , a role name r , and individual names a, b . An *ABox* is a finite set of such assertions. Terminological knowledge is expressed with *concept inclusions* (CIs) $C \sqsubseteq D$ composed of concept descriptions C, D , and a *TBox* is a finite set of CIs. An *ontology* is a pair of an ABox and a TBox.

\mathcal{EL} can be translated into first-order logic and thus has a model-theoretic semantics, which we define next. An *interpretation* \mathcal{I} consists of a *domain* $\text{Dom}(\mathcal{I})$, which is a non-empty set of *objects*, and of a function $\cdot^{\mathcal{I}}$ that gives meaning to the individual names a , concept names A , and role names r in the signature Σ by assigning them to elements $a^{\mathcal{I}}$, subsets $A^{\mathcal{I}}$, and binary relations $r^{\mathcal{I}}$, respectively, of $\text{Dom}(\mathcal{I})$. The interpretation function is extended to compound concepts by $\top^{\mathcal{I}} := \text{Dom}(\mathcal{I})$, $(C \sqcap D)^{\mathcal{I}} := C^{\mathcal{I}} \cap D^{\mathcal{I}}$, and $(\exists r.C)^{\mathcal{I}} := \{x \mid (x, y) \in r^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}} \text{ for some } y\}$. Simply put, \top describes the concept of all objects, $C \sqcap D$ describes the concept of all objects that are described by C as well as by D ,

and $\exists r.C$ describes the concept of all objects that are related by r to an object described by C . Moreover, \mathcal{I} satisfies a concept assertion $C(a)$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$ (the individual name a belongs to the concept described by C), a role assertion $r(a, b)$ if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$ (the role name r connects individual name a with the individual name b), and a concept inclusion $C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ (each object described by C is also described by D). We say that \mathcal{I} is a *model* of an ABox \mathcal{A} (a TBox \mathcal{T}), written $\mathcal{I} \models \mathcal{A}$ ($\mathcal{I} \models \mathcal{T}$), if \mathcal{I} satisfies all assertions in \mathcal{A} (all CIs in \mathcal{T}).

Reasoning is the process of deciding or enumerating consequences of an ontology. For instance, we say that a concept assertion $C(a)$ is *entailed* by an ABox \mathcal{A} w.r.t. a TBox \mathcal{T} if $C(a)$ is satisfied in all models of \mathcal{A} and \mathcal{T} ; this is abbreviated as $\mathcal{A} \models^{\mathcal{T}} C(a)$ and we also say that a is an *instance* of C w.r.t. \mathcal{A} and \mathcal{T} . Similarly, a CI $C \sqsubseteq D$ is *entailed* by \mathcal{T} if $C \sqsubseteq D$ is satisfied in every model of \mathcal{T} ; we then write $C \sqsubseteq^{\mathcal{T}} D$ and also say that C is *subsumed* by D w.r.t. \mathcal{T} . In \mathcal{EL} both reasoning problems can be decided in polynomial time by means of the Completion algorithm [5], which is implemented in the reasoner ELK [32].

2.1 Quantified ABoxes

ABoxes containing existential restrictions imply existence of anonymous individuals, e.g., in each model \mathcal{I} of $\{(\exists r.A)(a)\}$ there is an object y with $(a^{\mathcal{I}}, y) \in r^{\mathcal{I}}$ and $y \in A^{\mathcal{I}}$. In order to make such anonymous individuals explicit but also to allow more compact representations by employing structural sharing, we introduced quantified ABoxes. These are ABoxes that may use variables in addition to individual names and in which the concepts in assertions must not be compound. More precisely, a *quantified ABox* (*qABox*) $\exists X.\mathcal{A}$ consists of a finite set X of *variables*, which is disjoint with the signature Σ , and of a *matrix* \mathcal{A} , which is a finite set of assertions $A(u)$ and $r(u, v)$ where A is a concept name, r a role name, and u, v individual names or variables. An *object* of $\exists X.\mathcal{A}$ is either an individual name in Σ or a variable in X , and the set of all objects is denoted by $\text{Obj}(\exists X.\mathcal{A})$. An interpretation \mathcal{I} is a *model* of $\exists X.\mathcal{A}$, written $\mathcal{I} \models \exists X.\mathcal{A}$, if there is an assignment \mathcal{Z} that maps each variable x to an object $x^{\mathcal{Z}}$ in $\text{Dom}(\mathcal{I})$ such that the augmented interpretation $\mathcal{I}[\mathcal{Z}]$ is a model of the matrix \mathcal{A} . Entailment of assertions by qABoxes is defined as for ABoxes and is decidable in polynomial time too. Moreover, a qABox $\exists X.\mathcal{A}$ *entails* a qABox $\exists Y.\mathcal{B}$ w.r.t. a TBox \mathcal{T} if every model of $\exists X.\mathcal{A}$ and \mathcal{T} is also a model of $\exists Y.\mathcal{B}$, written $\exists X.\mathcal{A} \models^{\mathcal{T}} \exists Y.\mathcal{B}$. Entailment between qABoxes is NP-complete, even if $\mathcal{T} = \emptyset$ [19]. Although the Web Ontology Language OWL 2 [43] allows for anonymous individuals in ontologies and can thus represent qABoxes, they are not supported by the OWL 2 EL profile [40] since reasoning with variables is intractable.

Besides the model-based entailment $\models^{\mathcal{T}}$, we can also compare two qABoxes according to their consequences. One such consequence-based entailment considers all instance queries (which is a synonym for concept assertions): a qABox $\exists X.\mathcal{A}$ *IQ-entails* a qABox $\exists Y.\mathcal{B}$ w.r.t. \mathcal{T} , written $\exists X.\mathcal{A} \models_{\text{IQ}}^{\mathcal{T}} \exists Y.\mathcal{B}$, if $\exists Y.\mathcal{B} \models^{\mathcal{T}} C(a)$ implies $\exists X.\mathcal{A} \models^{\mathcal{T}} C(a)$ for every $C(a)$. In addition to concept assertions we can also take all role assertions into account, yielding *IRQ-entailment* $\models_{\text{IRQ}}^{\mathcal{T}}$. Another consequence-based entailment considers all Boolean conjunctive queries

(BCQs). Since BCQs and qABoxes are equivalent formalisms, we can define that $\exists X.\mathcal{A}$ CQ-entails $\exists Y.\mathcal{B}$ w.r.t. \mathcal{T} , written $\exists X.\mathcal{A} \models_{\text{CQ}}^{\mathcal{T}} \exists Y.\mathcal{B}$, if $\exists Y.\mathcal{B} \models^{\mathcal{T}} \exists Z.\mathcal{C}$ implies $\exists X.\mathcal{A} \models^{\mathcal{T}} \exists Z.\mathcal{C}$ for every $\exists Z.\mathcal{C}$. Since the TBox is fixed, CQ-entailment $\models_{\text{CQ}}^{\mathcal{T}}$ and model-based entailment $\models^{\mathcal{T}}$ coincide. However, IRQ-entailment $\models_{\text{IRQ}}^{\mathcal{T}}$ is strictly weaker, and IQ-entailment $\models_{\text{IQ}}^{\mathcal{T}}$ is the weakest of them.

Every ABox can be transformed into an equivalent qABox, but the converse is not true. For instance, the ABox $\{(A \sqcap \exists r.B)(a)\}$ can be rewritten to $\exists\{x\}. \{A(a), r(a, x), B(x)\}$. It is obtained by introducing the variable x as an r -successor of a for the anonymous individual in $(\exists r.B)(a)$. In the converse direction, the qABox $\exists\{y\}. \{s(a, y), C(y), r(y, y)\}$ has no equivalent ABox as it entails the infinitely many concept assertions $(\exists s. \dots \exists s.C)(a)$, but also since the cycle $r(y, y)$ between variables cannot be represented in an ABox. The situation may be different with another entailment. Specifically, w.r.t. the TBox $\{C \sqsubseteq \exists r.C\}$ the latter qABox is IRQ-equivalent to $\{(\exists s.C)(a)\}$. In general, whether a qABox is IRQ-equivalent to an ABox w.r.t. some TBox can be decided in polynomial time and, if so, such an ABox can be computed in exponential time [11].

2.2 Simulations and Homomorphisms

The consequence-based entailments have structural characterizations by means of simulations and homomorphisms, respectively. $\exists X.\mathcal{A}$ IQ-entails $\exists Y.\mathcal{B}$ w.r.t. the empty TBox if there is a *simulation* from $\exists Y.\mathcal{B}$ to $\exists X.\mathcal{A}$, which is a relation $\mathfrak{S} \subseteq \text{Obj}(\exists Y.\mathcal{B}) \times \text{Obj}(\exists X.\mathcal{A})$ that fulfills the following conditions:

- (S1) If a is an individual name, then $(a, a) \in \mathfrak{S}$.
- (S2) If $(u, u') \in \mathfrak{S}$ and $A(u) \in \mathcal{B}$, then $A(u') \in \mathcal{A}$.
- (S3) If $(u, u') \in \mathfrak{S}$ and $r(u, v) \in \mathcal{B}$, then $(v, v') \in \mathfrak{S}$ and $r(u', v') \in \mathcal{A}$ for some v' .

For a non-empty TBox \mathcal{T} we need to consider the IQ-saturation $\text{sat}_{\text{IQ}}^{\mathcal{T}}(\exists X.\mathcal{A})$ of the first qABox, obtained by materializing consequences implied by the TBox. To this end, we exhaustively apply the following rule, which terminates in polynomial time.

IQ-Saturation Rule. Choose an object u of $\exists X.\mathcal{A}$ as well as a CI $C \sqsubseteq D$ in \mathcal{T} with $\mathcal{A} \models C(u)$ but $\mathcal{A} \not\models D(u)$, and return the qABox obtained from $\exists X.\mathcal{A}$ by IQ-unfolding D at u , where “IQ-unfolding E at v ” is a recursive operation that does the following:

1. For each concept name $A \in \text{Conj}(E)$, add the assertion $A(v)$ to \mathcal{A} .
2. For each existential restriction $\exists r.F \in \text{Conj}(E)$, add the variable x_F to X , add the assertion $r(v, x_F)$ to \mathcal{A} , and IQ-unfold F at x_F .

Then, $\exists X.\mathcal{A} \models_{\text{IQ}}^{\mathcal{T}} \exists Y.\mathcal{B}$ iff there is a simulation from $\exists Y.\mathcal{B}$ to $\text{sat}_{\text{IQ}}^{\mathcal{T}}(\exists X.\mathcal{A})$. To decide IRQ-entailment $\models_{\text{IRQ}}^{\mathcal{T}}$ one additionally needs to check whether all role assertions in $\exists Y.\mathcal{B}$ involving individual names only are also contained in $\exists X.\mathcal{A}$.

Note that IQ-saturations are very similar to the *canonical models* constructed by the rule-based Completion algorithm [4], which is implemented in the reasoner ELK [32]. It can thus be used to efficiently compute IQ-saturations.

Furthermore, a *homomorphism* is a function $h: \text{Obj}(\exists Y.\mathcal{B}) \rightarrow \text{Obj}(\exists X.\mathcal{A})$ for which the relation $\{(u, h(u)) \mid u \in \text{Obj}(\exists Y.\mathcal{B})\}$ is a simulation. It holds that $\exists X.\mathcal{A} \models_{\text{CQ}}^{\mathcal{T}} \exists Y.\mathcal{B}$ iff there is a homomorphism from $\exists Y.\mathcal{B}$ to $\text{sat}_{\text{CQ}}^{\mathcal{T}}(\exists X.\mathcal{A})$, but to ensure finiteness of the CQ-saturation $\text{sat}_{\text{CQ}}^{\mathcal{T}}(\exists X.\mathcal{A})$ the TBox \mathcal{T} must be cycle-restricted, see [9] for details. The latter means that there is no concept description C and no role names r_1, \dots, r_n with $C \sqsubseteq^{\mathcal{T}} \exists r_1. \dots \exists r_n. C$ [2].

To construct the CQ-saturation, we replace the IQ-Saturation Rule with the variant for CQ. The difference to the IQ-Saturation Rule is that we now need to record the path on which we reached a particular object, and that we cannot reuse variables created from the same subconcepts.

CQ-Saturation Rule. Choose an object u of $\exists X.\mathcal{A}$ as well as a CI $C \sqsubseteq D$ in \mathcal{T} with $\mathcal{A} \models C(u)$ but $\mathcal{A} \not\models D(u)$, and return the qABox obtained from $\exists X.\mathcal{A}$ by unfolding D at u , where “unfolding E at p ” is a recursive operation that does the following:

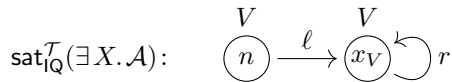
1. For each concept name $A \in \text{Conj}(E)$, add the assertion $A(p)$ to \mathcal{A} .
2. For each existential restriction $\exists r.F \in \text{Conj}(E)$, add the variable $p \xrightarrow{r} x_F$ to X , add the assertion $r(p, p \xrightarrow{r} x_F)$ to \mathcal{A} , and unfold F at $p \xrightarrow{r} x_F$.

Exhaustively applying the CQ-Saturation Rule terminates in exponential time for cycle-restricted TBoxes. For other TBoxes it need not terminate.

CQ-saturations correspond to structures often employed for conjunctive query answering over databases w.r.t. sets of existential rules, namely the *universal model* and the *chase* [27]. Termination of the chase is not decidable but there exist numerous sufficient conditions for termination [35]. For this close relationship, CQ-saturations can efficiently be computed with the reasoner VLog [25].

We will illustrate central notions with a running example, starting below.

Example 1. The input ontology consists of the qABox $\exists X.\mathcal{A} := \exists \emptyset. \{V(n)\}$ and the \mathcal{EL} TBox $\mathcal{T} := \{V \sqsubseteq \exists \ell.V, \exists \ell.V \sqsubseteq V\}$ (see introduction). Since the TBox is not cycle-restricted, no finite CQ-saturation exists, but we will see in Section 5.4 how an infinite CQ-saturation can be constructed in a well-defined way. We can, however, already here construct the IQ-saturation. Since the individual n is an instance of the premise of the CI $V \sqsubseteq \exists \ell.V$ but not of its conclusion, we extend the qABox such that n also instantiates the conclusion. We therefore introduce one fresh variable x_V , which corresponds to the anonymous individual in $\exists \ell.V$, together with the assertions $\ell(n, x_V)$ and $V(x_V)$. Since now also x_V is an instance of the same premise but not the conclusion, we need to extend the qABox again but now reuse the variable x_V , so that we add the assertion $\ell(x_V, x_V)$ only. Afterwards, the IQ-Saturation Rule is not applicable anymore and we are done. We have obtained $\text{sat}_{\text{IQ}}^{\mathcal{T}}(\exists X.\mathcal{A}) = \exists \{x_V\}. \{V(n), \ell(n, x_V), V(x_V), \ell(x_V, x_V)\}$, which is graphically represented below.



2.3 A Rewrite System for qABox Entailment

In order to further ease understanding the three different entailment relations between qABoxes, we provide yet another characterization by means of rewrite systems. We start with the easiest, which characterizes model-based entailment and CQ-entailment. There are three rules: the Copy Rule copies an object into a fresh variable, the Deletion Rule deletes an object or an assertion, and the CQ-Saturation Rule adds assertions implied by the TBox (see the previous section).

Copy Rule. Choose an object u of $\exists X.\mathcal{A}$ as well as a fresh variable y , i.e., $y \notin \text{Obj}(\exists X.\mathcal{A})$, and return the qABox $\exists(X \cup \{y\}).(\mathcal{A} \cup \{A(y) \mid A(u) \in \mathcal{A}\} \cup \{r(t, y) \mid r(t, u) \in \mathcal{A}\} \cup \{r(y, y) \mid r(u, u) \in \mathcal{A}\} \cup \{r(y, v) \mid r(u, v) \in \mathcal{A}\})$.

Delete Rule. Choose an assertion α in \mathcal{A} and return the qABox $\exists X.(\mathcal{A} \setminus \{\alpha\})$, or choose an object u of $\exists X.\mathcal{A}$ and return the qABox $\exists(X \setminus \{u\}).\{\alpha \mid \alpha \in \mathcal{A} \text{ and } u \text{ does not occur in } \alpha\}$.

Proposition 2. $\exists X.\mathcal{A} \models_{\text{CQ}}^{\mathcal{T}} \exists Y.\mathcal{B}$ iff $\exists X.\mathcal{A}$ can be rewritten to $\exists Y.\mathcal{B}$ by means of the three rules (Copy Rule, Delete Rule, CQ-Saturation Rule).

The above proposition holds for arbitrary TBoxes, but here we can only give a proof for cycle-restricted TBoxes. In order to formulate a proof for the general case, we first need to develop a notion of a CQ-saturation. The general proof is then given in Section 5.5.

Proof. Regarding the if direction, one first shows that a single rule application produces a qABox that is entailed, and then the claim follows by induction along the sequence of rule applications.

We proceed with the only-if direction, but restrict attention to a cycle-restricted TBox \mathcal{T} . Assume that $\exists X.\mathcal{A} \models^{\mathcal{T}} \exists Y.\mathcal{B}$, i.e., there is a homomorphism h from $\exists Y.\mathcal{B}$ to the CQ-saturation $\text{sat}_{\text{CQ}}^{\mathcal{T}}(\exists X.\mathcal{A})$. Recall that the latter can be constructed in exponential time by means of the CQ-Saturation Rule.

We will construct a sequence of qABoxes $\exists Z_i.\mathcal{C}_i$ such that each qABox $\exists Z_{i+1}.\mathcal{C}_{i+1}$ is obtained from the previous qABox $\exists Z_i.\mathcal{C}_i$ by means of the Copy Rule and the Delete Rule, accompanied with homomorphisms h_i from $\exists Y.\mathcal{B}$ to $\exists Z_i.\mathcal{C}_i$. The sequence starts with $\exists Z_0.\mathcal{C}_0 := \text{sat}_{\text{CQ}}^{\mathcal{T}}(\exists X.\mathcal{A})$ and $h_0 := h$ from above, and will end with a qABox and accompanying homomorphism that is bijective.

First of all, we use the Delete Rule to remove all objects (including all their assertions) to which the initial homomorphism h_0 does not map. Afterwards, the homomorphism is surjective. Injectivity is gained by repeatedly searching for pre-images $h_i^{-1}(v) := \{u \mid h_i(u) = v\}$ that contain more than one object: if $h_i^{-1}(v) = \{u_1, \dots, u_n\}$ where $n > 1$ and w.l.o.g. u_2, \dots, u_n are variables,⁶ then we use the Copy Rule to create $n - 1$ duplicates v_2, \dots, v_n of v (which yields the next qABox $\exists Z_{i+1}.\mathcal{C}_{i+1}$), and we obtain the next homomorphism h_{i+1} from h_i by keeping $h_{i+1}(v) = u_1$ but redefining $h_{i+1}(v_j) := u_j$ for each $j \in \{2, \dots, n\}$.

⁶ Since homomorphisms send each individual to itself, a pre-image $h_i^{-1}(v)$ cannot contain more than one individual.

The final homomorphism h_ℓ is bijective since each pre-image contains exactly one object. In order to obtain an isomorphism, we use the Delete Rule to remove all assertions that are not in the image of the homomorphism. Last, we use the Copy Rule and the Delete Rule to rename each variable $h_\ell(x)$ to x . The so obtained qABox equals $\exists Y.\mathcal{B}$. \square

Also, IQ-entailment can be characterized by a rewrite system. It uses the Copy Rule, the Delete Rule, and the Saturation Rule as well, but of course needs the IQ-Saturation Rule in place of the variant for CQ. However, these three rules are not enough. For instance, the qABox $\exists\{x\}.\{A(a), r(a, x)\}$ IQ-entails the qABox $\exists\{y\}.\{r(a, b), B(y)\}$. To see this, note that $A(a)$ and $(\exists r.\top)(a)$ are all non-trivial, atomic instance queries entailed by the first qABox, whereas the second qABox only entails $(\exists r.\top)(a)$. Since the former cannot yet be rewritten to the latter, we need to extend the rewrite system with further rules.

In the example, we see that the role assertion $r(a, x)$ is replaced by $r(a, b)$. This is possible since x and b are instances of the same \mathcal{EL} concept descriptions (namely of \top only), and thus no further fillers C in implied instance queries $(\exists r.C)(a)$ are introduced. In general, IQ-entailment is ignorant of the second object in role assertions — it can only detect whether there is a role assertion of the form $r(a, ?)$, namely by means of the instance query $(\exists r.\top)(a)$, but it cannot refer to the second individual in it.

In other words, the role assertions themselves need not bear meaning, but they are only used to describe the concepts of which the individuals are instances of. By rule of thumb: when IQ-entailment is employed, then one should not directly look into the qABoxes, but one must only use instance queries to access the knowledge they contain. We conclude that IQ-entailment must not be used in applications where connections between individuals, expressed by role assertions, are important. One should then rather use IRQ- or CQ-entailment.

Let us return to our endeavor of formulating additional rewrite rules. As we have learned above, whenever we find two objects u and u' that are instances of the same concept descriptions, then we can make u an r -successor of every object that has u' as r -successor, and vice versa. By doing so, no additional implied instance queries are introduced. A single-sided variant of this rule is as follows.

Simulated Successors Rule. Choose two objects u and u' of $\exists X.\mathcal{A}$ with $u \preceq u'$, and return the qABox $\exists X.(\mathcal{A} \cup \{r(t, u) \mid r(t, u') \in \mathcal{A}\})$.

In general, for two objects $u \in \text{Obj}(\exists X.\mathcal{A})$ and $u' \in \text{Obj}(\exists Y.\mathcal{B})$, the condition $u \preceq u'$ means that, for all \mathcal{EL} concept descriptions C , if $\mathcal{A} \models C(u)$, then $\mathcal{B} \models C(u')$. It is a finger exercise to show the following.

Lemma 3. $u \preceq u'$ iff (u, u') is contained in a simulation from $\exists X.\mathcal{A}$ to $\exists Y.\mathcal{B}$.

The following technical lemma is later used to guarantee subsequent applicability of the Simulated Successors Rule.

Lemma 4. If $\exists X.\mathcal{A}$ is a qABox on which \mathfrak{S} is a transitive simulation containing (u, u') , and the qABox $\exists Y.\mathcal{B}$ is obtained by applying the Simulated Successors Rule for (u, u') , then \mathfrak{S} is also a simulation on $\exists Y.\mathcal{B}$.

Proof. It is trivial that \mathfrak{S} still satisfies (S1) and (S2). We proceed with verifying (S3), where the only interesting case considers a pair $(t, t') \in \mathfrak{S}$ and an added role assertion $r(t, u) \in \mathcal{B} \setminus \mathcal{A}$. Recall that then $(u, u') \in \mathfrak{S}$ and $r(t, u') \in \mathcal{A}$. (S3) yields an object u'' with $(u', u'') \in \mathfrak{S}$ and $r(t', u'') \in \mathcal{A}$. For \mathfrak{S} being transitive, we conclude that $(u, u'') \in \mathfrak{S}$. \square

In our example, the IQ-entailed qABox further contains the concept assertion $B(y)$ but the first qABox does not contain any corresponding assertion. In fact, it does not even contain the concept name B . The reason is that IQ-entailment is ignorant of assertions that are not reachable from individuals, simply because it cannot access such assertions by means of instance queries. Therefore, we also need a rule to add arbitrary such assertions.

Unreachable Assertions Rule. Choose a fresh variable y and return the qABox $\exists(X \cup \{y\}).\mathcal{A}$, or choose a variable y in X that is not reachable from any individual and choose a fresh assertion α involving y such that, if α is a role assertion, then the first object is neither an individual nor reachable from any individual, and return the qABox $\exists X.(\mathcal{A} \cup \{\alpha\})$.

We now verify that the five aforementioned rules yield a sound and complete characterization of IQ-entailment.

Proposition 5. $\exists X.\mathcal{A} \models_{\text{IQ}}^{\mathcal{T}} \exists Y.\mathcal{B}$ iff $\exists X.\mathcal{A}$ can be rewritten to $\exists Y.\mathcal{B}$ by means of the five rules (Copy Rule, Delete Rule, IQ-Saturation Rule, Simulated Successors Rule, Unreachable Assertions Rule).

Proof. Regarding the if direction, one first shows that a single rule application produces a qABox that is IQ-entailed, and then the claim follows by induction along the sequence of rule applications.

We proceed with the only-if direction and therefore assume that $\exists X.\mathcal{A} \models_{\text{IQ}}^{\mathcal{T}} \exists Y.\mathcal{B}$. We will construct a finite sequence of qABoxes $\exists X_i.\mathcal{A}_i$ that starts with $\exists X_0.\mathcal{A}_0 := \exists X.\mathcal{A}$ and such that $\exists X_{i+1}.\mathcal{A}_{i+1}$ is obtained from $\exists X_i.\mathcal{A}_i$ by means of the five rules (Copy Rule, Delete Rule, IQ-Saturation Rule, Simulated Successors Rule, Unreachable Assertions Rule). Every subsequent qABox $\exists X_i.\mathcal{A}_i$ with $i \geq 1$ will be accompanied with a simulation \mathfrak{S}_i from $\exists Y.\mathcal{B}$ to $\exists X_i.\mathcal{A}_i$ such that the last simulation is an isomorphism.

Before we start with constructing the sequence, we need to explain when a simulation is an isomorphism. In general, we say that a relation $R \subseteq B \times A$ from a set B to a set A is

- *left-total* if, for each $b \in B$, there is at least one $a \in A$ with $(b, a) \in R$,
- *right-definite* or *functional* if, for each $b \in B$, there is at most one $a \in A$ with $(b, a) \in R$,
- *right-total* or *surjective* if, for each $a \in A$, there is at least one $b \in B$ with $(b, a) \in R$,
- *left-definite* or *injective* if, for each $a \in A$, there is at most one $b \in B$ with $(b, a) \in R$,

Each left-total, functional relation R induces the unique function $f: B \rightarrow A$ with $f(b) := a$ if $(b, a) \in R$. Slightly abusing notation, we identify R and f in this case and say that R is a function. Further recall that each injective, surjective function f has an inverse $f^{-1}: A \rightarrow B$ with $f^{-1}(a) := b$ if $f(b) = a$.

If a simulation \mathfrak{S} from $\exists Y.\mathcal{B}$ to $\exists X.\mathcal{A}$ is a function, then it is even a homomorphism from $\exists Y.\mathcal{B}$ to $\exists X.\mathcal{A}$. An isomorphism is a homomorphism that is injective and surjective, and the inverse of which is a homomorphism in the converse direction. Quantified ABoxes between which there is an isomorphism are equal up to renaming of variables.

In the following, we will use the five rewrite rules to first construct the saturation of $\exists X.\mathcal{A}$ (Step 1) and then construct qABoxes such that the accompanying simulation is surjective (Step 2), injective (Step 3), functional (Steps 4 and 5), and left-total (Step 6). In the final Step 7, we will delete superfluous assertions to ensure that the obtained injective, surjective homomorphism is also an isomorphism, and we then rename the variables to establish equality between the rewritten qABox and the input qABox $\exists Y.\mathcal{B}$.

1. The qABox $\exists X_1.\mathcal{A}_1$ is defined as the saturation $\text{sat}_{\text{IQ}}^{\mathcal{T}}(\exists X.\mathcal{A})$, which can be constructed from $\exists X_0.\mathcal{A}_0 = \exists X.\mathcal{A}$ with the IQ-Saturation Rule. Since $\exists X.\mathcal{A} \models_{\text{IQ}}^{\mathcal{T}} \exists Y.\mathcal{B}$, there is a simulation \mathfrak{S} from $\exists Y.\mathcal{B}$ to $\text{sat}_{\text{IQ}}^{\mathcal{T}}(\exists X.\mathcal{A})$. As first simulation \mathfrak{S}_1 we take this \mathfrak{S} .
2. To obtain $\exists X_2.\mathcal{A}_2$, we use the Delete Rule to remove all objects from $\exists X_1.\mathcal{A}_1$ that do not occur in \mathfrak{S}_1 . We do not need to modify the simulation and therefore set $\mathfrak{S}_2 := \mathfrak{S}_1$. This simulation \mathfrak{S}_2 from $\exists Y.\mathcal{B}$ to $\exists X_2.\mathcal{A}_2$ is surjective.
3. Next, we construct the qABox $\exists X_3.\mathcal{A}_3$ so that the simulation \mathfrak{S}_3 is injective. For each pair $(u, u') \in \mathfrak{S}_2$ except where u and u' are the same individual, we use the Copy Rule to create a duplicate of u' , which we denote by $x_{(u, u')}$. In the following, we treat a and $x_{(a, a)}$ as synonyms for each individual a . Also, we use the Delete Rule to remove every variable u' with $(u, u') \in \mathfrak{S}_2$. The resulting qABox is denoted as $\exists X_3.\mathcal{A}_3$. The accompanying simulation \mathfrak{S}_3 is obtained from \mathfrak{S}_2 by replacing each pair (u, u') with $(u, x_{(u, u')}$, i.e., formally we define $\mathfrak{S}_3 := \{ (u, x_{(u, u')}) \mid (u, u') \in \mathfrak{S}_2 \}$. We verify that \mathfrak{S}_3 is a simulation from $\exists Y.\mathcal{B}$ to $\exists X_3.\mathcal{A}_3$.
 - (S1) Consider an individual a . (S1) yields $(a, a) \in \mathfrak{S}_2$. So $(a, x_{(a, a)}) \in \mathfrak{S}_3$. Recall that a and $x_{(a, a)}$ are synonyms, and thus $(a, a) \in \mathfrak{S}_3$.
 - (S2) Let $(u, x_{(u, u')}) \in \mathfrak{S}_3$, i.e., $(u, u') \in \mathfrak{S}_2$. Further consider a concept assertion $A(u) \in \mathcal{B}$. (S2) yields $A(u') \in \mathcal{A}_2$. Recall that $x_{(u, u')}$ is either a duplicate of u' created by the Copy Rule or a synonym of the individual $u = u'$, and thus $A(x_{(u, u')}) \in \mathcal{A}_3$.
 - (S3) Again let $(u, x_{(u, u')}) \in \mathfrak{S}_3$, i.e., $(u, u') \in \mathfrak{S}_2$, but now consider a role assertion $r(u, v) \in \mathcal{B}$. (S3) yields an object v' with $(v, v') \in \mathfrak{S}_2$ and $r(u', v') \in \mathcal{A}_2$. Recall that $x_{(u, u')}$ is either a duplicate of u' created by the Copy Rule or a synonym of the individual $u = u'$, and similarly for $x_{(v, v')}$, and thus $r(x_{(u, u')}, x_{(v, v')}) \in \mathcal{A}_3$.

4. In preparation of Step 5, we construct a qABox $\exists X_4. \mathcal{A}_4$ so that, for each two pairs (u, u') and (u, u'') in the corresponding simulation \mathfrak{S}_4 , the objects u' and u'' are instances of the same \mathcal{EL} concept descriptions, i.e., $u' \preceq u''$ and $u'' \preceq u'$.

For each pair $(u, u') \in \mathfrak{S}_3$, we use the Delete Rule to delete superfluous assertions involving u' , i.e., which are not needed to simulate u . Since \mathfrak{S}_3 is injective, this removal can be done without considering objects other than u . Formally, we obtain from $\exists X_3. \mathcal{A}_3$ the qABox with same variables, i.e., $X_4 := X_3$, and with the matrix

$$\begin{aligned} \mathcal{A}_4 := & \mathcal{A}_3 \setminus \{ A(u') \mid (u, u') \in \mathfrak{S}_3 \text{ and } A(u) \notin \mathcal{B} \} \\ & \setminus \{ r(u', v') \mid (u, u') \in \mathfrak{S}_3 \text{ and } r(u, v) \notin \mathcal{B} \text{ for each } (v, v') \in \mathfrak{S}_3 \}. \end{aligned}$$

We first show that \mathfrak{S}_3 is also a simulation from $\exists Y. \mathcal{B}$ to $\exists X_4. \mathcal{A}_4$.

(S1) We already know that $(a, a) \in \mathfrak{S}_3$ for each individual name a .

(S2) Let $(u, u') \in \mathfrak{S}_3$ and $A(u) \in \mathcal{B}$. Since \mathfrak{S}_3 fulfills (S2) it follows that $A(u') \in \mathcal{A}_3$. According to the definition of \mathcal{A}_4 and since \mathfrak{S}_3 is injective, this assertion $A(u')$ is not removed and thus $A(u') \in \mathcal{A}_4$.

(S3) Consider $(u, u') \in \mathfrak{S}_3$ and $r(u, v) \in \mathcal{B}$. By (S3) there is v' with $(v, v') \in \mathfrak{S}_3$ and $r(u', v') \in \mathcal{A}_3$. By definition of \mathcal{A}_4 and for \mathfrak{S}_3 is injective, the assertion $r(u', v')$ is not removed, i.e., $r(u', v') \in \mathcal{A}_4$.

Next, we verify that the inverse relation $\mathfrak{S}_3^{-1} := \{ (u', u) \mid (u, u') \in \mathfrak{S}_3 \}$ is a simulation from $\exists X_4. \mathcal{A}_4$ to $\exists Y. \mathcal{B}$.

(S1) If a is an individual, then $(a, a) \in \mathfrak{S}_3$ by (S1) and thus $(a, a) \in \mathfrak{S}_3^{-1}$.

(S2) Let $(u', u) \in \mathfrak{S}_3^{-1}$ and $A(u') \in \mathcal{A}_4$. Then $(u, u') \in \mathfrak{S}_3$. If $A(u)$ was not in \mathcal{B} , then \mathcal{A}_4 would not contain $A(u')$, a contradiction.

(S3) Last, assume $(u', u) \in \mathfrak{S}_3^{-1}$ and $r(u', v') \in \mathcal{A}_4$. Then $(u, v) \in \mathfrak{S}_3$. If $r(u, v)$ was not in \mathcal{B} for each $(v', v) \in \mathfrak{S}_3^{-1}$, then \mathcal{A}_4 would not contain $r(u', v')$, a contradiction.

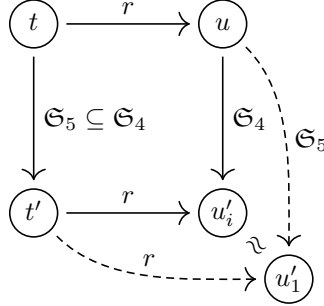
Since simulations are closed under composition, the relation $\mathfrak{S}_3^{-1} \circ \mathfrak{S}_3$ is a simulation on $\exists X_4. \mathcal{A}_4$, as is the transitive closure of $\mathfrak{S}_3^{-1} \circ \mathfrak{S}_3$ which we denote by \mathfrak{T}_4 . Since, for each two pairs (u, u') and (u, u'') in \mathfrak{S}_3 , the pairs (u', u'') and (u'', u') are in $\mathfrak{S}_3^{-1} \circ \mathfrak{S}_3$ and thus also in \mathfrak{T}_4 , we conclude that $u' \preceq u''$ and $u'' \preceq u'$. As next simulation we define $\mathfrak{S}_4 := \mathfrak{S}_3$.

5. As next step, we construct a qABox $\exists X_5. \mathcal{A}_5$ for which the accompanying simulation \mathfrak{S}_5 is functional.

One after another, we consider every object u of $\exists Y. \mathcal{B}$. First of all, we fix an enumeration $\{u'_1, \dots, u'_n\}$ of the set $\{u' \mid (u, u') \in \mathfrak{S}_4\}$. If u is an individual name, then $(u, u) \in \mathfrak{S}_4$ and so we let $u'_1 = u$. For \mathfrak{S}_4 satisfies (S1) and is injective, the other objects u'_i with $i \geq 2$ are variables. Recall from the last step that $(u'_i, u'_j) \in \mathfrak{T}_4$ for all indexes i, j , and thus we can apply the Simulated Successors Rule for each pair (u'_1, u'_i) with $i \geq 2$. Note that Lemma 4 ensures that the rule remains applicable. After all objects of $\exists Y. \mathcal{B}$ have been considered, we use the Delete Rule to remove the variables u'_2, \dots, u'_n for every $u \in \text{Obj}(\exists Y. \mathcal{B})$. The resulting qABox is denoted by $\exists X_5. \mathcal{A}_5$.

We verify that the relation \mathfrak{S}_5 , obtained from \mathfrak{S}_4 by removing each pair (u, u'_i) with $i \geq 2$, is a simulation from $\exists Y. \mathcal{B}$ to $\exists X_5. \mathcal{A}_5$.

- (S1) If u is an individual name, then we chose $u = u'_1$ above. Since the pair (u, u'_1) is not removed, it is contained in \mathfrak{S}_5 .
- (S2) Since $\mathfrak{S}_4 \supseteq \mathfrak{S}_5$, (S2) is still fulfilled by each pair in \mathfrak{S}_5 .
- (S3) Consider a role assertion $r(t, u) \in \mathcal{B}$ and let $(t, t') \in \mathfrak{S}_5$. Since $\mathfrak{S}_5 \subseteq \mathfrak{S}_4$, it follows that $(t, t') \in \mathfrak{S}_4$ and so (S3) yields $r(t', u'_i) \in \mathcal{A}_4$ for some $(u, u'_i) \in \mathfrak{S}_4$. Further note that t' was not deleted and is still an object of $\exists X_5. \mathcal{A}_5$, since $(t, t') \in \mathfrak{S}_5$ implies that t' is the first object t'_1 in the enumeration of $\{t' \mid (t, t') \in \mathfrak{S}_4\}$. Since the Simulated Successors Rule was applied to (u'_1, u'_i) , it follows that $r(t', u'_1) \in \mathcal{A}_5$. Obviously, we have $(u, u'_1) \in \mathfrak{S}_5$ by definition.



6. The simulation \mathfrak{S}_5 might not yet be left-total and thus no function. The reason is that there might be objects of $\exists Y. \mathcal{B}$ that do not occur in any pair in \mathfrak{S}_5 , since they are not reachable from an individual. We therefore use the Unreachable Assertions Rule to copy over those parts from $\exists Y. \mathcal{B}$, and denote the resulting qABox by $\exists X_6. \mathcal{A}_6$. Furthermore, we obtain the simulation \mathfrak{S}_6 from \mathfrak{S}_5 by extending it with the pairs (x, x) for all added variables x .
7. Finally, the last simulation \mathfrak{S}_6 is a function that is injective and surjective, and thus a bijective homomorphism, which we denote by h . To ensure that its inverse is also a homomorphism, we use the Delete Rule to remove all assertions from $\exists X_6. \mathcal{A}_6$ that are not in the image of the homomorphism h . Thus, the resulting qABox and the input qABox $\exists Y. \mathcal{B}$ are equal up to renaming of variables. Finally, we use the Copy Rule and the Delete Rule to rename each variable $h(x)$ to x . The so obtained qABox $\exists X_7. \mathcal{A}_7$ equals $\exists Y. \mathcal{B}$. \square

In order to further adapt the rewrite system to IRQ-entailment, we only need to change the Simulated Successors Rule such that it does not add new role assertions involving only individuals.

We now need to reconsider Step 5 in the above proof. Assume that t and u'_1 are individuals, i.e., $u'_1 = u$. Specifically, when the modified Simulated Successors Rule is applied for a pair (u'_1, u'_i) , the role assertion $r(t, u'_1)$ will not be added anymore for any role assertion $r(t, u'_i)$ in \mathcal{A}_4 . It seems that we cannot delete the variable u' , and can thus not achieve that the simulation is functional. However, this is wrong. To see this, we make a case distinction.

- First, let $r(t, u)$ be contained in the target qABox $\exists Y. \mathcal{B}$. Since $\exists Y. \mathcal{B}$ is IRQ-entailed by $\exists X. \mathcal{A}$, this assertion $r(t, u)$ must also be contained in $\exists X. \mathcal{A}$. It

will not be deleted in any step and is thus contained in every intermediate qABox $\exists X_k. \mathcal{A}_k$. Thus, the needed assertion $r(t, u'_1)$ is already there and thus need not be added in Step 5.

- Otherwise, if $\exists Y. \mathcal{B}$ does not contain $r(t, u)$, then it is unproblematic when the intermediate qABox $\exists X_5. \mathcal{A}_5$ (at the end of Step 5) does not contain this assertion either — the relation \mathfrak{S}_5 is then still a simulation.

3 A Brief Recap of Optimal Repairs

The framework of optimal repairs has been developed in a series of conference articles [9, 11, 13, 14, 19, 34]. We will recall and explain the main results for the case where the data is represented by a qABox $\exists X. \mathcal{A}$ and there is a static \mathcal{EL} TBox \mathcal{T} that describes terminological knowledge about the domain of interest. In applications a reasoner such as ELK [32] is employed to derive consequences from $\exists X. \mathcal{A}$ and \mathcal{T} . When it is detected that an individual a is an instance of a concept C , i.e., $\exists X. \mathcal{A} \models^{\mathcal{T}} C(a)$, and this consequence is deemed to be incorrect, then the qABox needs to be repaired. In this situation, we prefer to compute an *optimal* repair, which is a qABox that is entailed by $\exists X. \mathcal{A}$ and \mathcal{T} , does not entail the unwanted consequence $C(a)$, and entails as many other consequences as possible.

More formally, we assume that the *repair request* \mathcal{P} is a finite set of \mathcal{EL} concept assertions (the unwanted consequences).⁷ A *repair* of $\exists X. \mathcal{A}$ for \mathcal{P} w.r.t. \mathcal{T} is a qABox $\exists Y. \mathcal{B}$ with $\exists X. \mathcal{A} \models^{\mathcal{T}} \exists Y. \mathcal{B}$ and $\exists Y. \mathcal{B} \not\models^{\mathcal{T}} C(a)$ for each $C(a) \in \mathcal{P}$. It is *optimal* if it is not strictly entailed by another repair, i.e., $\exists Y. \mathcal{B} \models^{\mathcal{T}} \exists Y'. \mathcal{B}'$ for each other repair $\exists Y'. \mathcal{B}'$ with $\exists Y'. \mathcal{B}' \models^{\mathcal{T}} \exists Y. \mathcal{B}$. For a query language $QL \in \{\text{CQ}, \text{IRQ}, \text{IQ}\}$, the definition of (optimal) QL-repairs is obtained by replacing the model-based entailment $\models^{\mathcal{T}}$ with the consequence-based entailment $\models_{QL}^{\mathcal{T}}$. Recall that $\models^{\mathcal{T}}$ and $\models_{\text{CQ}}^{\mathcal{T}}$ coincide, and thus repairs and CQ-repairs are the same.

Since every (optimal) repair is entailed by the input qABox, we can construct one by means of the rewrite system from Section 2.3. However, this will often be very inefficient for the huge size of the search space. From the investigations of neighbors of \mathcal{EL} concept descriptions [33] it follows that one sequence of subsequent rewritten of the qABox $\exists \{x_1, \dots, x_n\}. \{r_1(a, x_1), r_2(x_1, x_2), \dots, r_n(x_{n-1}, x_n), A_1(x_n), \dots, A_k(x_n)\}$ can contain n -fold exponentially many qABoxes, up to equivalence.

Another problem is that the rewrite system is not terminating: there is no upper bound on the number of applications of the Copy Rule and, depending on the TBox, also of the CQ-Saturation Rule. If the TBox is cycle-restricted [2], then a *bounded repair property* can be shown with the filtration technique [13]. Every repair is then entailed by an exponentially-large repair. With that, the rewrite system is turned into a terminating one: the rules that create new objects (Copy Rule and Saturation Rule) can only be applied if the number of objects does not yet exceed the bound. As an alternative, we could also switch to IQ- or IRQ-entailment for which the bounded repair property is satisfied for all TBoxes.

⁷ The symbol \mathcal{R} is usually reserved for RBoxes, so we use \mathcal{P} to avoid clashes.

Goal-directed applications of the rewrite rules can be planned by means of the *repair recipe*, obtained as negation of the recursive characterization of $\exists X. \mathcal{A} \models C(a)$ (first with empty TBox). Specifically: to make \mathcal{A} not entail $C(u)$, one can either choose a concept name $B \in \text{Conj}(C)$ and remove $B(u)$ from \mathcal{A} or choose an existential restriction $\exists r.D \in \text{Conj}(C)$ and, for each $r(u, v)$ in \mathcal{A} , either recursively make \mathcal{A} not entail $D(v)$ or remove $r(u, v)$ from \mathcal{A} . The maximal amount of consequences is preserved if first enough copies are created with the Copy Rule such that afterwards the Delete Rule is applied according to the repair recipe in every possible way. If the TBox is empty, then every optimal repair can be constructed in this way.

Example 6. As an example, consider the qABox $\exists \{x\}. \{has_topping(my_pizza, x), Salami(x), Parmesan(x)\}$. Since nothing can be both salami and parmesan, it has the unwanted consequence $(\exists has_topping.(Salami \sqcap Parmesan))(my_pizza)$.

We first apply the repair recipe without prior copying. We need to make the matrix not entail $(\exists has_topping.(Salami \sqcap Parmesan))(my_pizza)$. Since $\exists has_topping.(Salami \sqcap Parmesan)$ is an atom, it is the only top-level conjunct of itself, i.e., we can only choose this atom and then recursively apply the repair recipe. To this end, we consider all *has_topping*-successors of *my_pizza*, of which there is only one, namely *x*. Then, we need to make the matrix not entail $(Salami \sqcap Parmesan)(x)$, and we can therefore choose one of the top-level conjuncts *Salami* or *Parmesan*, i.e., we remove either *Salami(x)* or *Parmesan(x)*. We end up with a pizza that has either salami or parmesan as topping.

If we initially copy the variable *x* into a fresh variable *y*, then we can apply the repair recipe to *x* and *y* differently, and end up with the optimal repair $\exists \{x, y\}. \{has_topping(my_pizza, x), Salami(x), has_topping(my_pizza, y), Parmesan(y)\}$, in which the pizza still has both salami and parmesan as toppings (but no single topping anymore that is both salami and parmesan).

The TBox is taken into account by forward and backward chaining. On the one hand, we initially use the Saturation Rule to materialize consequences implied by the TBox. We can then keep those that do not violate the repair request, and thereby retain all consequences that could only be inferred using assertions that will be removed. On the other hand, we must prevent that the TBox can be used to restore removed consequences. More specifically, when we apply the repair recipe to make \mathcal{A} not entail $C(u)$ and thereby choose the atom $G \in \text{Conj}(C)$, and if $E \sqsubseteq F$ is a CI in \mathcal{T} where F is subsumed by G , then it is also mandatory to make \mathcal{A} not entail $E(u)$. Otherwise, inferencing with the TBox (specifically with $E \sqsubseteq F$) would restore that u is an instance of G and thus also of C .

All above ideas culminated in the definition of the *canonical repairs*. Each of them is induced by a so-called repair seed, which specifies which atomic consequences are to be removed. First of all, a *repair seed* \mathcal{S} maps each individual name a to a set \mathcal{S}_a of concept descriptions. The intended meaning is that, in the repair, a is no instance of any concept in \mathcal{S}_a anymore. Such a seed can be chosen as follows, resembling the above repair recipe.

We initialize \mathcal{S} by adding the concept C to \mathcal{S}_a for each unwanted consequence $C(a)$ in \mathcal{P} , but only for those entailed by $\exists X.\mathcal{A}$ and \mathcal{T} (otherwise there is no need to repair). Then we exhaustively apply the following three rules to \mathcal{S} .

1. If \mathcal{S}_a contains a conjunction C , then we choose an atom G from $\text{Conj}(C)$ and add G to \mathcal{S}_a .
2. If \mathcal{S}_a contains an existential restriction $\exists r.D$ and $\exists X.\mathcal{A}$ contains the role assertion $r(a, b)$ where b is an instance of D , then we either add D to \mathcal{S}_b (in which case the repair will entail $r(a, b)$ but not $D(b)$ anymore) or do nothing (then the repair will entail $D(b)$ but not $r(a, b)$ anymore).
3. If \mathcal{S}_a contains an atom G and the TBox \mathcal{T} contains a concept inclusion $E \sqsubseteq F$ with $F \sqsubseteq^\emptyset G$ and where a is an instance of E , then we add E to \mathcal{S}_a .

Finally, we only keep subsumption-maximal concepts in each set \mathcal{S}_a . This does not change the instructions how the repair should be built, because if $C \sqsubseteq^\emptyset D$ and $\exists Y.\mathcal{B}$ does not entail $D(a)$, then $\exists Y.\mathcal{B}$ must not entail $C(a)$ as well.

In the following we give the definition of the canonical IQ-repairs only; the results for the CQ-repairs are similar but will not be used in the sequel of this article because we are interested in the more general case where the TBox \mathcal{T} need not be cycle-restricted. Each canonical IQ-repair consists of all objects of the form $\langle u, \mathcal{K} \rangle$ where u is an object of the IQ-saturation and \mathcal{K} is a repair type.⁸ Essentially, $\langle u, \mathcal{K} \rangle$ is a copy of u that is weakened according to \mathcal{K} . More specifically, each repair type \mathcal{K} is a subset of the set of atoms occurring in \mathcal{P} or in \mathcal{T} , and $\langle u, \mathcal{K} \rangle$ is no instance of any atom in \mathcal{K} . For reasons of efficiency as well as to guarantee that a repair is obtained, each repair type \mathcal{K} must additionally satisfy the following three conditions, where $\exists Y.\mathcal{B}$ is the IQ-saturation of $\exists X.\mathcal{A}$:

- (RT1)** $\mathcal{B} \models C(u)$ for each atom $C \in \mathcal{K}$
- (RT2)** $C \not\sqsubseteq^\emptyset D$ for each two atoms C, D in \mathcal{K}
- (RT3)** If C is an atom in \mathcal{K} and $E \sqsubseteq F$ is a CI in \mathcal{T} with $\mathcal{B} \models E(u)$ and $F \sqsubseteq^\emptyset C$, then there is an atom D in \mathcal{K} such that $E \sqsubseteq^\emptyset D$.⁹

The matrix of each canonical IQ-repair consists of the following assertions:

- (CR1)** $A(\langle u, \mathcal{K} \rangle)$ if $A(u) \in \mathcal{B}$ and $A \notin \mathcal{K}$.
- (CR2)** $r(\langle u, \mathcal{K} \rangle, \langle v, \mathcal{L} \rangle)$ if $r(u, v) \in \mathcal{B}$ and, for each $\exists r.C \in \mathcal{K}$ with $\mathcal{B} \models C(v)$, there is an atom $D \in \mathcal{L}$ such that $C \sqsubseteq^\emptyset D$.

For the same input, the canonical IQ-repairs only differ in the selection which of the copies $\langle u, \mathcal{K} \rangle$ are identified with individual names. This selection is made by means of the repair seed \mathcal{S} , which we have already constructed above.¹⁰ By considering each individual name a and the copy $\langle a, \mathcal{S}_a \rangle$ as synonyms, we obtain

⁸ Instead of the notation $y_{u, \mathcal{K}}$ used in [9, 11, 19], we now write $\langle u, \mathcal{K} \rangle$ similar as in [13].

⁹ This condition is defined in a different way in [9]. However, it is only employed in Lemma XIII in [10] to show that the canonical repairs are already saturated, where this alternative condition already suffices.

¹⁰ In [9, 11, 19] we denoted a repair seed by s , but this often clashes with role names. Thus, we will instead use \mathcal{S} as in [13].

the canonical IQ-repair induced by \mathcal{S} , denoted by $\text{rep}_{\text{IQ}}^{\mathcal{T}}(\exists X.\mathcal{A}, \mathcal{S})$. Formally, a *repair seed* \mathcal{S} maps each individual name a to a repair type \mathcal{S}_a for a such that:

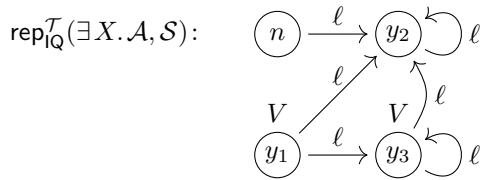
(RS) If $C(a)$ is an unwanted consequence in \mathcal{P} with $\mathcal{B} \models C(a)$, then there is an atom D in \mathcal{S}_a such that $C \sqsubseteq^{\emptyset} D$.

Up to IQ-equivalence, the set of all optimal IQ-repairs can be computed in exponential time. This complexity result cannot be improved since, in the worst cases, exponentially many optimal IQ-repairs can exist and optimal IQ-repairs can be of exponential size (and not be equivalent to sub-exponential qABoxes).

As pointed out in Section 2.3, one peculiarity of IQ-entailment is its ignorance of many role assertions. A strange example: the ABox $\{r(a, b), B(b)\}$ has, for the repair request $\{(\exists r. B)(a)\}$, the optimal IQ-repair $\{r(a, c), B(b)\}$, where a, b, c are individuals. The added role assertion $r(a, c)$ is itself not entailed by the given ABox, but is only used to retain the consequence $(\exists r. \top)(a)$. That is why IQ-repairs must not be used when the contained role assertions will be accessed directly. One should then rather switch to IRQ- or CQ-repairs.

Notably, our canonical IQ-repairs are not problematic in the above mentioned respect since new role assertions between individuals are not introduced and as many of the existing ones as possible are preserved, although this is not necessary for IQ-entailment. (The same need not hold for other IQ-repairs.) In particular, the canonical IQ-repairs are also IRQ-repairs, but they need to be compared differently, i.e., optimal IQ-repairs and optimal IRQ-repairs do not coincide [11].

Example 7. We continue our running example from Example 1. As repair request we take $\mathcal{P} := \{V(n)\}$. Since there are no conjunctions in the TBox \mathcal{T} or in the repair request \mathcal{P} , there is only one repair seed \mathcal{S} where $\mathcal{S}_n = \{V, \exists \ell. V\}$. Specifically, V must be contained in \mathcal{S}_n for the unwanted consequence $V(n)$ in \mathcal{P} , and also the atom $\exists \ell. V$ must be in \mathcal{S}_n for the CI $\exists \ell. V \sqsubseteq V$ in \mathcal{T} of which the conclusion is subsumed by an atom already in \mathcal{S}_n (namely V itself). Since every optimal repair is equivalent to a canonical repair, and every canonical repair is induced by a repair seed, only one optimal repair exists here, up to equivalence. The repair seed \mathcal{S} induces the canonical IQ-repair $\text{rep}_{\text{IQ}}^{\mathcal{T}}(\exists X.\mathcal{A}, \mathcal{S})$, which is graphically represented below; n and $\langle n, \{V, \exists \ell. V\} \rangle$ are synonyms, and $y_1 := \langle n, \emptyset \rangle$, $y_2 := \langle x_V, \{V, \exists \ell. V\} \rangle$, $y_3 := \langle x_V, \emptyset \rangle$.



4 Concise Representations of Canonical IQ-Repairs

Canonical IQ-repairs are of exponential size, not only in the worst case, but also in the best case. In this section, we consider two approaches for alleviating this

problem. One approach produces considerably smaller repairs in practice, which may, however, still be exponential in the worst case. The second approach uses the polynomial-sized repair seeds as representations for the exponentially large canonical repairs.

4.1 Optimized IQ-repairs

To avoid generating exponential-sized repairs also in the best case, we have developed in [9] an optimized algorithm for computing repairs induced by repair seeds. Intuitively, these *optimized repairs* do not contain all the objects occurring in the canonical repair, but only those that are really needed. We have shown that the optimized IQ-repair induced by a repair seed \mathcal{S} is IQ-equivalent to the canonical one induced by \mathcal{S} , and thus the set of optimized IQ-repairs can be used in place of the set of canonical ones when computing the optimal repairs. The experiments described in [9] show that the optimized repairs are in most cases considerably smaller than the canonical ones. For instance, in the canonical IQ-repair we have just computed for our Narcissus example (see Example 7), the objects y_1 and y_3 are not needed since they are not reachable from n . IQ-equivalence of the optimized repair $\exists\{y_2\}.\{\ell(n, y_2), \ell(y_2, y_2)\}$ with the canonical one can be seen by using the identity on the objects n and y_2 as simulation in both directions.

Note, however, that in general an exponential blow-up cannot be avoided, as already shown in [14] for a restricted class of qABoxes without a TBox. This blow-up is not only a problem when computing the repair, but also when using it later on to answer queries. While answering IQs is polynomial for the original (unrepaired) qABox, it may become exponential after the repair if we measure the complexity in the size the repair problem, consisting of the original qABox, the TBox, and the repair request.

4.2 Representing canonical IQ-repairs by repair seeds

The size of a repair seed \mathcal{S} is polynomial in the size of the repair problem, and it uniquely determines the induced canonical repair $\text{rep}_{\text{IQ}}^{\mathcal{T}}(\exists X.\mathcal{A}, \mathcal{S})$. To take advantage of this more concise representation of canonical repairs, we must be able to work directly with this representation when comparing the repairs w.r.t. IQ-entailment and when answering IQs w.r.t. them. The following proposition shows how this can be realized.

Proposition 8 ([12, 15]). *Let \mathcal{T} be an \mathcal{EL} TBox, $\exists X.\mathcal{A}$ a qABox, \mathcal{R} a repair request, $\mathcal{S}, \mathcal{S}'$ repair seeds, and $E(b)$ an \mathcal{EL} concept assertion. Then,*

1. $\text{rep}_{\text{IQ}}^{\mathcal{T}}(\exists X.\mathcal{A}, \mathcal{S}) \models_{\text{IQ}}^{\mathcal{T}} \text{rep}_{\text{IQ}}^{\mathcal{T}}(\exists X.\mathcal{A}, \mathcal{S}')$ iff for each individual name a and for each atom $C \in \mathcal{S}_a$, there is an atom $D \in \mathcal{S}'_a$ with $C \sqsubseteq^{\emptyset} D$.
2. $\text{rep}_{\text{IQ}}^{\mathcal{T}}(\exists X.\mathcal{A}, \mathcal{S}) \models^{\mathcal{T}} E(b)$ iff $\exists X.\mathcal{A} \models^{\mathcal{T}} E(b)$ and \mathcal{S}_b does not contain any atom D with $E \sqsubseteq^{\mathcal{T}} D$.

Size Ontology				Size ABox				Size TBox			
min.	max.	med.	avg.	min.	max.	med.	avg.	min.	max.	med.	avg.
154	891,452	6,751	77,761.5	103	747,998	2,089	46,625.7	61	473,254	2,706	31,135.8

Table 1. Statistics of the used corpus of \mathcal{EL} ontologies after filtering out non- \mathcal{EL} axioms.

The conditions formulated in this proposition are clearly decidable in time polynomial in the size of the repair problem. Thus, from a theoretical point of view, representing canonical repairs using repair seeds is preferable to using optimized repairs since the worst-case complexity of the relevant inference problems is polynomial for the former, whereas it is exponential for the latter. Comparing the worst-case complexity of two algorithms does not always tell us which algorithm will perform better in practice. To investigate the advantages and disadvantages of our two concise representations of canonical IQ-repairs in practice, we performed experiments on real-world ontologies.

4.3 Experimental evaluation

The goal of the experiments was to evaluate the performance of the two representations with respect to the time needed for answering instance queries. To this end, we created a benchmark consisting of \mathcal{EL} ontologies, instance queries, and repair requests.

Ontologies As in the experiments in [9], which mainly compared the sizes of the optimized repairs with that of the canonical ones, we took the ontologies from the OWL EL Materialization track of the OWL Ontology Reasoner Evaluation 2015 [42], filtering out axioms that cannot be expressed in \mathcal{EL} . To test the limits of both approaches, we this time included all 109 ontologies from this corpus, instead of considering only ontologies of up to 100,000 axioms as in [9]. Table 1 provides information on how large the employed ontologies were.

Queries For each ontology, we randomly generated 100 instance queries which would return at least one individual name before repairing. For this, we employed the following probabilistic recursive procedure on the IQ-saturation $\text{sat}_{\text{IQ}}^T(\mathcal{A})$ of the ABox \mathcal{A} . We initialize *the current object* u with a randomly selected individual name from \mathcal{A} , the *current concept* C with \top , and then use the following procedure to generate a random concept description C s.t. $\mathcal{A} \models^T C(u)$.

1. If there exists a role r and an object v for the current object u s.t. $r(u, v) \in \text{sat}_{\text{IQ}}^T(\mathcal{A})$, then, with a 50% probability, select a random such pair (r, v) , recursively call this procedure to generate a random concept description D for v , and add $\exists r.D$ as conjunct to the current concept description C .
2. If no conjunct was added in the previous step, and there exists a concept name A s.t. $A(u) \in \text{sat}_{\text{IQ}}^A(\mathcal{A})$, select a random such concept name A and add it as a conjunct to C .

3. With a 50% probability, return the current concept description C , and otherwise continue with Step 1.

Repair Requests and Seed Functions We furthermore generated two sets of repair requests for each ontology, **RR1** and **RR2**. The repair requests for **RR1** were generated using the approach employed in [9], which generates requests where the concept assertions involve only concept names. In addition, we this time also generated repair requests containing assertions with compound concept descriptions, which we denote by **RR2**.

For **RR1**, we randomly selected 50% of the individual names, to which we randomly assigned 10% of the concept names that the individual was an instance of. This was not possible for 14 ontologies, in which no individual was an instance of any concept name, and which were thus excluded from the experiment. The decision that repair requests are built from concept names only is justified by the fact that those concept names often stand for compound concept descriptions defined in the TBox, and represent central concepts of the ontology.

Using only concept names in the repair request also makes it easier to answer queries based on the seed function. We thus also generated a second set of repair requests **RR2** for each ontology that assigned randomly generated compound concept descriptions. For this, we used the same procedure as for generating instance queries, however this time using a bound of 5 on the concept size, measured as the number of symbols (logical connectives, as well as concept and role names) required to write the concept down. In particular, we only add conjuncts to the current concept description if the resulting concept does not extend the maximum size. This way, we assigned to each of 50% of randomly selected individual names 10 repair requests consisting of concepts of size at most 5. The repair requests generated in this way are denoted by **RR2**. Using both methods, we attempted to compute 10 seed functions per ontology based on the generated repair requests. We used a timeout of 10 minutes for this. We were successful in computing seed functions in 89.5% of cases for **RR1** (not counting the excluded ontologies), and in 72.5% of cases for **RR2** (which we attempted to generate for all 109 ontologies). Note that while the repair requests used for **RR1** did not contain compound concept descriptions, the seed functions did. Further note that we did not yet use the simplified approach to computing repair seeds from Page 18, but we believe that it is more efficient than the approach currently used in the prototype — we will investigate this in future experiments.

Results We computed optimized IQ repairs with a timeout of 1 hour, which was successful for 85.8% (**RR1**) / 100% (**RR2**) of seed functions on average (recall that for less **RR2** ontologies, the seed function computation was successful). Then we compared the performance of answering instance queries from the optimized repairs and from the repair seeds. Any required \mathcal{EL} reasoning was performed using ELK [32]. Figure 1 shows the results of this comparison, where each point corresponds to a tuple of ontology, repair request, and seed function, the x-axis to the run-time of evaluating all 100 instance queries using the repair seed, and

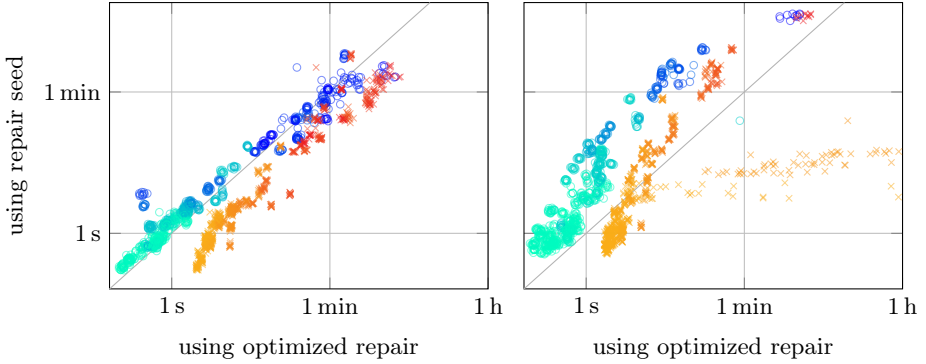


Fig. 1. Run times of evaluating 100 instance queries on repairs using the seed function (x-axis) vs. using the optimized repairs (y-axis). Color intensity corresponds to size of the input ontology. Orange-red crosses include times for computing the repair, whereas cyan-blue circles do not. Results of RR1 on the left, and for RR2 on the right.

the y-axis of evaluating all instance queries using the optimized repair, where the red color denotes that we also count the computation time of the optimized repair, and the blue color denotes that we do not. For RR1 with the simple repair requests, using the repair seed instead of the precomputed repair was faster in 98.7% of cases if we also count the time for computing the repair, and otherwise in 17.9% of cases. As we can see however in Figure 1, using the optimized repair was almost never significantly faster, and there were many cases in which using the repair seed instead of the repair was significantly faster even if we do not count the time for computing the repair. For RR2 with the complex repair requests, using the repair seed was faster in 64.6% of cases if we count the time for computing the repair, and otherwise almost never (0.13% of cases). The reason for this was that after obtaining the query answers from ELK, we still have to do a subsumption check for each individual in the answer when using the repair seed only (see the condition in Proposition 8). In RR2, each of these tests was more expensive, since we were comparing complex \mathcal{EL} concepts. When using the precomputed optimized repair, no additional subsumption tests are necessary.

The results show that computing the optimized repair explicitly rather than using the repair seed is only advisable if this repair is considered to be the final one, which is then used for many instance tests. This is not the case for intermediate repairs in a setting where the KE iteratively repairs the ontology by (a) choosing a repair seed, then (b) checking out the induced canonical repair by looking at some of its consequences, and based on this inspection deciding whether (c) to choose a different repair seed or (d) to use this repair seed, but maybe repair the obtained ontology further by formulating a new repair request. It then makes sense to compute the optimized repair only after the iterative repair process is finished.

If the repair is assumed to be the final one, a good indicator for when computing the optimized repair does not pay off is the size of the original ontology. If we consider RR1 and do not count the time for computing the repair, for ontologies with at most 404,509 axioms (85% of the corpus), using the repair seed was faster in only 6.8% of the cases, while for the larger ontologies, it was faster in 80.5% of the cases. The numbers are similar if we look at the size increase of the repair: if the repair contained at most 132,622 axioms more than the original ontology (85% of the corpus), then using the repair seed was faster in 5.5% of the cases, and otherwise in 87.5% of the cases.

5 Finite Representations of Optimal CQ-Repairs

Recall that the (finite) CQ-repairs in [9] can only be constructed w.r.t. cycle-restricted TBoxes. The goal of this section is to lift this severe restriction. We still assume that the input qABox is finite, which ensures that the rule-based IQ-saturation and also the canonical IQ-repairs exist and are finite. As repairs, however, we allow for finite or infinite qABoxes of arbitrary cardinality. By means of an operation called *shell unfolding*, we will transform the IQ-saturation into a CQ-saturation and, likewise, each canonical IQ-repair into a CQ-repair (which we call canonical). In effect, both the CQ-saturation and every canonical CQ-repair could become countably infinite.

We will further prove a compactness theorem for shell unfoldings, namely that model-based entailment and CQ-entailment are equal and coincide with existence of a homomorphism. As a generic result, we will show how existence of a homomorphism between shell unfoldings can be decided in non-deterministic polynomial time by only looking at the (finite) qABoxes from which the unfoldings are constructed. We conclude that CQ-entailment between two canonical CQ-repairs is decidable, even if the repairs are infinite. We further show that CQ-query answering w.r.t. a canonical CQ-repair is in NP (w.r.t. the size of the input only), which is the same computational complexity as without repairing. Thus each (possibly infinite) canonical CQ-repair has as finite representation the underlying canonical IQ-repair. Moreover, we verify that the set of all canonical CQ-repairs is complete, i.e., that every CQ-repair is CQ-entailed by a canonical one. As main result we show that (a representation of) the set of all optimal CQ-repairs can be computed in exponential time with access to an NP-oracle.

5.1 Infinite qABoxes

First of all, we extend the definition of a qABox $\exists X.\mathcal{A}$ such that both the variable set X and the matrix \mathcal{A} can be infinite (and of arbitrary cardinality). Infinite qABoxes cannot be translated into first-order logic. This raises the question which of the basic results on finite qABoxes still hold for the infinite ones. A key lemma was that model-based entailment corresponds to existence of a homomorphism. We will generalize this to the infinite qABoxes.

Given a qABox $\exists X.\mathcal{A}$, we define its *canonical model* as the interpretation $\text{CMod}(\exists X.\mathcal{A}) := \mathcal{I}$ with domain $\text{Dom}(\mathcal{I}) := \text{Obj}(\exists X.\mathcal{A})$ and function $\cdot^{\mathcal{I}}$ where $a^{\mathcal{I}} := a$ for each individual name a , and $A^{\mathcal{I}} := \{u \mid A(u) \in \mathcal{A}\}$ for each concept name A , and $r^{\mathcal{I}} := \{(u, v) \mid r(u, v) \in \mathcal{A}\}$ for each role name r .

Lemma 9. *Consider qABoxes $\exists X.\mathcal{A}$ and $\exists Y.\mathcal{B}$ with arbitrary cardinality. The following statements are equivalent.*

1. $\exists X.\mathcal{A} \models \exists Y.\mathcal{B}$
2. $\text{CMod}(\exists X.\mathcal{A}) \models \exists Y.\mathcal{B}$
3. *There is a homomorphism from $\exists Y.\mathcal{B}$ to $\exists X.\mathcal{A}$.*
4. *There is a variable assignment $\sigma: Y \rightarrow \text{Obj}(\exists X.\mathcal{A})$ with $\sigma(\mathcal{B}) \subseteq \mathcal{A}$, where $\sigma(\mathcal{B})$ is obtained from \mathcal{B} by replacing every variable y with $\sigma(y)$.*

Proof. Actually, the proof is the same as with finite qABoxes or for containment of conjunctive queries [26], but we will include it below so that all readers can convince themselves.

We denote by \mathcal{I} the canonical model $\text{CMod}(\exists X.\mathcal{A})$.

1 \Rightarrow 2. Let $\exists X.\mathcal{A} \models \exists Y.\mathcal{B}$. With the variable assignment $\mathcal{Z}: X \rightarrow \text{Obj}(\exists X.\mathcal{A})$ where $x^{\mathcal{Z}} := x$ for each $x \in X$, the augmented interpretation $\mathcal{I}[\mathcal{Z}]$ is a model of the matrix \mathcal{A} . It follows that the canonical model \mathcal{I} is a model of $\exists X.\mathcal{A}$, and thus $\mathcal{I} \models \exists Y.\mathcal{B}$.

2 \Rightarrow 3. Let \mathcal{I} be a model of $\exists Y.\mathcal{B}$, i.e., there is a variable assignment $\mathcal{W}: Y \rightarrow \text{Obj}(\exists X.\mathcal{A})$ such that the augmented interpretation $\mathcal{I}[\mathcal{W}]$ is a model of the matrix \mathcal{B} . We define the mapping $h: \text{Obj}(\exists Y.\mathcal{B}) \rightarrow \text{Obj}(\exists X.\mathcal{A})$ with $h(u) := u^{\mathcal{I}[\mathcal{W}]}$ for each $u \in \text{Obj}(\exists Y.\mathcal{B})$, and show that h is a homomorphism from $\exists Y.\mathcal{B}$ to $\exists X.\mathcal{A}$.

- For each individual name a , we have $h(a) = a^{\mathcal{I}[\mathcal{W}]} = a^{\mathcal{I}} = a$.
- If $A(u)$ is a concept assertion in \mathcal{B} , then $u^{\mathcal{I}[\mathcal{W}]} \in A^{\mathcal{I}}$, and thus $A(h(u)) \in \mathcal{A}$ by definition of \mathcal{I} .
- And if $r(u, v)$ is a role assertion in \mathcal{B} , then $(u^{\mathcal{I}[\mathcal{W}]}, v^{\mathcal{I}[\mathcal{W}]}) \in r^{\mathcal{I}}$, and thus $r(h(u), h(v)) \in \mathcal{A}$ by definition of \mathcal{I} .

3 \Rightarrow 1. Assume that there is a homomorphism h from $\exists Y.\mathcal{B}$ to $\exists X.\mathcal{A}$, and further let \mathcal{J} be a model of $\exists X.\mathcal{A}$, i.e., there is a variable assignment $\mathcal{Z}: X \rightarrow \text{Dom}(\mathcal{J})$ such that $\mathcal{J}[\mathcal{Z}]$ is a model of \mathcal{A} . We define the variable assignment $\mathcal{W}: Y \rightarrow \text{Dom}(\mathcal{J})$ with $y^{\mathcal{W}} := h(y)^{\mathcal{J}[\mathcal{Z}]}$ for each variable $y \in Y$. Then $u^{\mathcal{J}[\mathcal{W}]} = h(u)^{\mathcal{J}[\mathcal{Z}]}$ for each object u of $\exists Y.\mathcal{B}$. We will show that $\mathcal{J}[\mathcal{W}]$ is a model of \mathcal{B} , from which we conclude that \mathcal{J} is a model of $\exists Y.\mathcal{B}$.

- Consider a concept assertion $A(u)$ in \mathcal{B} . Since h is a homomorphism, \mathcal{A} contains $A(h(u))$. It follows that $h(u)^{\mathcal{J}[\mathcal{Z}]} \in A^{\mathcal{J}}$, and thus $u^{\mathcal{J}[\mathcal{W}]} \in A^{\mathcal{J}}$.
- In a similar way, if $r(u, v)$ is a role assertion in \mathcal{B} , then $r(h(u), h(v))$ is in \mathcal{A} , and thus $(h(u)^{\mathcal{J}[\mathcal{Z}]}, h(v)^{\mathcal{J}[\mathcal{Z}]}) \in r^{\mathcal{J}}$, i.e., $(u^{\mathcal{J}[\mathcal{W}]}, v^{\mathcal{J}[\mathcal{W}]}) \in r^{\mathcal{J}}$.

3 \Leftrightarrow 4. The difference between a homomorphism in Statement 3 and a variable assignment in Statement 4 is that the homomorphism must additionally send each individual to itself, otherwise they are the same.

- If h is a homomorphism from $\exists Y.\mathcal{B}$ to $\exists X.\mathcal{A}$, then we obtain the desired variable assignment σ by defining $\sigma(y) := h(y)$ for each $y \in Y$.
- If σ is a variable assignment with $\sigma(\mathcal{B}) \subseteq \mathcal{A}$, then we get the needed homomorphism h by defining $h(a) := a$ for each $a \in \Sigma_1$ and $h(y) := \sigma(y)$ for each $y \in Y$. \square

The above characterization does not take a TBox into account. In Section 5.4 we will generalize the case for finite qABoxes and cycle-restricted TBoxes: we will show how a saturation of a finite qABox $\exists X.\mathcal{A}$ w.r.t. an arbitrary \mathcal{EL} TBox \mathcal{T} can be constructed in a well-defined way such that, for each qABox $\exists Y.\mathcal{B}$ of arbitrary cardinality, $\exists X.\mathcal{A} \models^{\mathcal{T}} \exists Y.\mathcal{B}$ iff the saturation entails $\exists Y.\mathcal{B}$, i.e., iff there is a homomorphism from $\exists Y.\mathcal{B}$ to the saturation.

In a special case, we can already here provide the following characterization.

Definition 10. *We say that a qABox $\exists X.\mathcal{A}$ is saturated w.r.t. an \mathcal{EL} TBox \mathcal{T} if, for each CI $C \sqsubseteq D$ in \mathcal{T} and for each object u of $\exists X.\mathcal{A}$, if $\mathcal{A} \models C(u)$, then $\mathcal{A} \models D(u)$.*

Lemma 11. *Let $\exists X.\mathcal{A}$ and $\exists Y.\mathcal{B}$ be qABoxes of arbitrary cardinality, and let \mathcal{T} be an \mathcal{EL} TBox. Further assume that $\exists X.\mathcal{A}$ is saturated w.r.t. \mathcal{T} . Then $\exists X.\mathcal{A} \models^{\mathcal{T}} \exists Y.\mathcal{B}$ iff $\exists X.\mathcal{A} \models \exists Y.\mathcal{B}$.*

Proof. The if direction is trivial. Regarding the only-if direction, assume $\exists X.\mathcal{A} \models^{\mathcal{T}} \exists Y.\mathcal{B}$. As shown in the proof of Lemma 9, $\text{CMod}(\exists X.\mathcal{A})$ is a model of $\exists X.\mathcal{A}$. Since $\exists X.\mathcal{A}$ is saturated w.r.t. \mathcal{T} , $\text{CMod}(\exists X.\mathcal{A})$ is also a model of \mathcal{T} . We conclude that $\text{CMod}(\exists X.\mathcal{A})$ is a model of $\exists Y.\mathcal{B}$. Another application of Lemma 9 yields that $\exists X.\mathcal{A} \models \exists Y.\mathcal{B}$. \square

Existence of a simulation is still sufficient for IQ-entailment between infinite qABoxes, but the following example shows that it is not necessary. However, there are certain classes of infinite qABoxes where it is still necessary, e.g., if $\exists X.\mathcal{A}$ has finite out-degree¹¹ [38].¹² Analogously, the if direction in Lemma 3 also holds for all infinite qABoxes of arbitrary cardinality, but in general the only-if direction does not hold anymore (again, it holds if $\exists X.\mathcal{A}$ has finite out-degree).

Example 12. Consider the following two countably infinite qABoxes. The first qABox $\exists X.\mathcal{A}$ has variables $X := \{x_M \mid M \subseteq \mathbb{N} \text{ and } M \text{ is finite}\}$ and its matrix \mathcal{A} is the union of all $\{r(a, x_M)\} \cup \{B_i(x_M) \mid i \in M\}$ where $x_M \in X$. For every finite subset M of \mathbb{N} , the individual name a has an r -successor x_M that is an instance of all concept names B_i with $i \in M$. The second qABox is

¹¹ That is, for each object u of $\exists X.\mathcal{A}$, there are only finitely many assertions $r(u, v)$ in \mathcal{A} . This condition is satisfied by all shell unfoldings of finite qABoxes, see Section 5.2.

¹² See also the proof of Proposition 23 in [17].

$\exists Y. \mathcal{B} := \exists \{y\}. (\{r(a, y)\} \cup \{B_i(y) \mid i \in \mathbb{N}\})$. The individual name a has a single r -successor y , which is an instance of all concept names B_1, B_2, \dots ¹³

The atomic instance queries implied by $\exists Y. \mathcal{B}$ are $\top(a)$ and $(\exists r. \prod \{B_i \mid i \in M\})(a)$ for every finite subset M of \mathbb{N} . These are all implied by $\exists X. \mathcal{A}$ as well. Although $\exists X. \mathcal{A} \models_{\text{IQ}} \exists Y. \mathcal{B}$ (and thus also $a \preceq a$), there is no simulation from $\exists Y. \mathcal{B}$ to $\exists X. \mathcal{A}$ (which contains (a, a)).

The above example further shows that existence of a homomorphism is not necessary for CQ-entailment (although it is still sufficient): $\exists X. \mathcal{A} \models_{\text{CQ}} \exists Y. \mathcal{B}$ but no homomorphism from $\exists Y. \mathcal{B}$ to $\exists X. \mathcal{A}$ exists. Another example is as follows.

Example 13. As left-hand side, we consider the qABox representing the natural numbers with their usual order relation: $\exists X. \mathcal{A}$ with variables $X := \mathbb{N}$ and matrix $\mathcal{A} := \{r(m, n) \mid m < n\}$. As right-hand side, we take the real numbers: $\exists Y. \mathcal{B}$ with variables $Y := \mathbb{R}$ and matrix $\mathcal{B} := \{r(x, y) \mid x < y\}$. The former is countable and the latter is uncountable.

Each finite qABox entailed by $\exists Y. \mathcal{B}$ is also entailed by $\exists X. \mathcal{A}$, i.e., $\exists X. \mathcal{A} \models_{\text{CQ}} \exists Y. \mathcal{B}$. However, there is no homomorphism from $\exists Y. \mathcal{B}$ to $\exists X. \mathcal{A}$, for the larger cardinality of \mathbb{R} . No mapping from \mathbb{R} (the objects of $\exists Y. \mathcal{B}$) to \mathbb{N} (the objects of $\exists X. \mathcal{A}$) is injective. Thus, there are always two real numbers x, y (where w.l.o.g. $x < y$) that are mapped to the same natural number n , but then the role assertion $r(x, y)$ in \mathcal{B} has no homomorphic image since $r(n, n)$ is not in \mathcal{A} .

Thus contrary to finite qABoxes, for infinite qABoxes model-based entailment $\models^{\mathcal{T}}$ and CQ-entailment $\models_{\text{CQ}}^{\mathcal{T}}$ are not the same.

5.2 Shell Unfoldings and Homomorphisms

Consider a (finite) quantified ABox $\exists X. \mathcal{A}$, the objects of which are divided into *kernel objects* and *shell objects*, such that each individual name is a kernel object, each shell object is reachable from some kernel object, but no kernel object is reachable from any shell object. A *shell path* is a sequence $u_0 \xrightarrow{r_1} u_1 \xrightarrow{r_2} \dots \xrightarrow{r_n} u_n$ that starts with a kernel object u_0 but otherwise only contains shell objects u_1, \dots, u_n such that \mathcal{A} contains $r_i(u_{i-1}, u_i)$ for all $i \in \{1, \dots, n\}$. We call $n \geq 0$ its *length*, u_0 its *source*, and u_n its *target*. Note that kernel objects, and thus also individuals, can be seen as shell paths of length 0. The target of such a shell path representing a kernel object is this object itself.

Definition 14. *The shell unfolding of $\exists X. \mathcal{A}$ is defined as the quantified ABox $\exists X'. \mathcal{A}'$ with the following components:*

$$X' := \{p \mid p \text{ is a shell path where } p \notin \Sigma_1\},$$

¹³ To use a finite signature only, we can encode each concept name B_i (which represents the number i) as the existential restriction $(\exists s_1.)^i \exists s_2. \top$ where s_1 and s_2 are two role names different from r . These are all incomparable w.r.t. subsumption, just like the concept names B_i .

$$\begin{aligned} \mathcal{A}' := & \{ A(p) \mid p \text{ is a shell path with target } u \text{ and } A(u) \in \mathcal{A} \} \cup \\ & \{ r(u, v) \mid u, v \text{ are kernel objects and } r(u, v) \in \mathcal{A} \} \cup \\ & \{ r(p, q) \mid p, q \text{ are shell paths such that } q = p \xrightarrow{r} u \text{ for a shell object } u \}. \end{aligned}$$

Each shell unfolding is either finite or countably infinite. Such infinite qABoxes cannot be directly used in computing devices, but as shell unfoldings they are finitely represented by the underlying qABox $\exists X.\mathcal{A}$ and the partitioning into kernel objects and shell objects. In the sequel of this section we will investigate how entailment involving shell unfoldings can be decided by means of these finite representations. To this end, we exploit that a shell unfolding is “almost everywhere tree-shaped,” i.e., the role assertions involving all but finitely many objects induce a directed graph that is a tree. Furthermore, this tree-shaped part is regular in the sense that it can be recognized by a finite-state automaton.

It is easy to see that a given qABox CQ-entails its shell unfolding, but the entailment in the other direction holds only w.r.t. IQ-entailment.

Proposition 15. *Let $\exists X.\mathcal{A}$ be a finite qABox and $\exists X'.\mathcal{A}'$ its shell unfolding.*

1. *The mapping $h: p \mapsto u$ where u is the target of the shell path p is a homomorphism from $\exists X'.\mathcal{A}'$ to $\exists X.\mathcal{A}$, and thus $\exists X.\mathcal{A}$ CQ-entails $\exists X'.\mathcal{A}'$.*
2. *The relation $\mathfrak{S} := \{ (u, p) \mid u \text{ is the target of the shell path } p \}$ is a simulation from $\exists X.\mathcal{A}$ to $\exists X'.\mathcal{A}'$, and thus $\exists X'.\mathcal{A}'$ IQ-entails $\exists X.\mathcal{A}$.*

In general, the shell unfolding need not CQ-entail the original qABox.

Example 16. Consider the qABox $\exists X.\mathcal{A} := \exists \{x\}. \{r(a, x), r(x, x)\}$, of which the individual a is the only kernel object. Assume that g is a homomorphism from $\exists X.\mathcal{A}$ to its shell unfolding $\exists X'.\mathcal{A}'$, and let $g(x) = p$ for a shell path p . Since $r(x, x) \in \mathcal{A}$, we must have $r(g(x), g(x)) \in \mathcal{A}'$. However, the only r -successor of $g(x) = p$ in \mathcal{A}' is $p \xrightarrow{r} x$, which is different from p . Another way of seeing why $\exists X.\mathcal{A}$ is not CQ-entailed by $\exists X'.\mathcal{A}'$ is to observe that the conjunctive query $\exists y.r(y, y)$ is entailed by $\exists X.\mathcal{A}$, but not by $\exists X'.\mathcal{A}'$.

Given two finite qABoxes $\exists X.\mathcal{A}$ and $\exists Y.\mathcal{B}$ whose object sets are partitioned into kernel objects and shell objects as introduced above, we are interested in finding out whether there exists a homomorphism from the shell unfolding $\exists X'.\mathcal{A}'$ of $\exists X.\mathcal{A}$ into the shell unfolding $\exists Y'.\mathcal{B}'$ of $\exists Y.\mathcal{B}$, which is the same as checking whether $\exists Y'.\mathcal{B}'$ CQ-entails $\exists X'.\mathcal{A}'$. This is a non-trivial problem since the shell unfoldings may be infinite, and thus we cannot just compute the unfoldings and then try to guess a homomorphism between them. The following lemma gives a necessary and sufficient condition for this.

Lemma 17. *There is a homomorphism from $\exists X'.\mathcal{A}'$ to $\exists Y'.\mathcal{B}'$ iff there is a simulation from $\exists X.\mathcal{A}$ to $\exists Y.\mathcal{B}$ whose restriction to the kernel objects of $\exists X.\mathcal{A}$ is a function.*

Proof. Assume that g is a homomorphism from $\exists X'.\mathcal{A}'$ to $\exists Y'.\mathcal{B}'$, which is of course also a simulation. By Proposition 15, there is a simulation \mathfrak{S} from $\exists X.\mathcal{A}$

to $\exists X'.\mathcal{A}'$ that is the identity on the kernel objects of $\exists X.\mathcal{A}$. In fact, if u is a kernel object, then the only shell path that has u as its target is u itself. The composition of \mathfrak{S} with g is thus a simulation from $\exists X.\mathcal{A}$ to $\exists Y'.\mathcal{B}'$ that coincides with the function g on the kernel objects of $\exists X.\mathcal{A}$. This shows the only-if direction of the proposition.

For the if direction, assume that \mathfrak{T} is a simulation from $\exists X.\mathcal{A}$ to $\exists Y'.\mathcal{B}'$ whose restriction to the kernel objects of $\exists X.\mathcal{A}$ is a function h . We extend h to shell paths of length $n > 0$ by induction on n . First, consider a shell path $q = u \xrightarrow{\tau} v$ of length 1. Then u is a kernel object, v is a shell object, and $r(u, v) \in \mathcal{A}$. Let $u' := h(u)$. Since $(u, u') \in \mathfrak{T}$ and $r(u, v) \in \mathcal{A}$, there is an object v' in \mathcal{B}' such that $r(u', v') \in \mathcal{B}'$ and $(v, v') \in \mathfrak{T}$. We set $h(q) := v'$ for an arbitrary such object v' . For the induction step, let $q = p \xrightarrow{\tau} v$ be a shell path of length $n > 1$, i.e., p is a shell path of length $n - 1$ with target u , v is a shell object, and $r(u, v) \in \mathcal{A}$. By induction, we can assume that $u' := h(p)$ is already defined, and satisfies $(u, u') \in \mathfrak{T}$. Thus, $r(u, v) \in \mathcal{A}$ implies that there is an object v' in \mathcal{B}' such that $r(u', v') \in \mathcal{B}'$ and $(v, v') \in \mathfrak{T}$. Again, we set $h(q) := v'$ for an arbitrary such object v' .

It remains to show that the function h from the objects of $\exists X'.\mathcal{A}'$ to the objects of $\exists Y'.\mathcal{B}'$ defined this way is indeed a homomorphism. First, note that every $a \in \Sigma_1$ is a kernel object, and thus we have $h(a) = a$ since $(a, a) \in \mathfrak{T}$ and h is the restriction of \mathfrak{T} to the kernel objects of $\exists X.\mathcal{A}$ by assumption. Second, assume that the shell path q in the objects of $\exists X'.\mathcal{A}'$ satisfies $A(q) \in \mathcal{A}'$, which means that $A(u) \in \mathcal{A}$ for the target u of q . By our construction of h , we know that $(u, h(q)) \in \mathfrak{T}$, and thus $A(u) \in \mathcal{A}$ implies $A(h(q)) \in \mathcal{B}'$, as required. Third, assume that $r(u, v) \in \mathcal{A}'$ for kernel objects u, v . Then $r(u, v) \in \mathcal{A}$ and the fact that h coincides with the simulation \mathfrak{T} on the kernel objects of $\exists X.\mathcal{A}$ implies $r(h(u), h(v)) \in \mathcal{B}'$. Finally, assume that $r(p, q) \in \mathcal{A}'$ for shell paths p, q . Then $q = p \xrightarrow{\tau} v$ for a shell object v and $r(u, v) \in \mathcal{A}$, where u is the target of p . By our construction of h , we know that $(h(p), h(q)) \in \mathcal{B}'$, which finishes our proof that h is a homomorphism. \square

From a more abstract point of view, the if direction of this lemma holds since the non-kernel part of \mathcal{A}' consists of trees, and it is well-known that there exists a homomorphism from a tree into a graph iff there exists a simulation from the tree to the graph. One advantage of this characterization is that the source of the simulation is the finite qABox $\exists X.\mathcal{A}$ rather than its infinite shell unfolding $\exists X'.\mathcal{A}'$. However, the target $\exists Y'.\mathcal{B}'$ is still infinite. The next lemma shows that one does not have to look very deeply into the unfolded part of \mathcal{B}' when trying to construct the functional part of the simulation.

Lemma 18. *If there is a simulation from $\exists X.\mathcal{A}$ to $\exists Y'.\mathcal{B}'$ whose restriction to the kernel objects of $\exists X.\mathcal{A}$ is a function, then there is such a simulation that additionally satisfies that the images of the kernel objects are shell paths whose length is bounded by $k + \ell$, where k is number of kernel objects of $\exists X.\mathcal{A}$ and ℓ is the number of shell objects of $\exists Y.\mathcal{B}$.*

Proof. Let \mathfrak{S} be a simulation from $\exists X.\mathcal{A}$ to $\exists Y'.\mathcal{B}'$ whose restriction to the kernel objects of $\exists X.\mathcal{A}$ is the function h . We now consider the graph obtained as the image under h of the kernel objects of $\exists X.\mathcal{A}$, and partition this graph into its connected components, where edges are used in a bi-directional way when defining connectedness.

First, consider a connected component that contains a kernel object u of $\exists Y.\mathcal{B}$, and let p be a shell path of length > 0 in this component. Since p is (undirected) reachable from u and shell objects cannot reach kernel objects in a directed way, there must be a kernel object v of $\exists Y.\mathcal{B}$ such that p has v as its source. Assume that p is of the form $p = v \xrightarrow{r_1} u_1 \dots \xrightarrow{r_n} u_n$, and let $p_i = v \xrightarrow{r_1} u_1 \dots \xrightarrow{r_i} u_i$ be the initial segment of p of length i , for $i = 0, \dots, n$. Then the unique path in the connected component with which $p = p_n$ is reached from $v = p_0$ uses the role assertions $r_1(v, p_1), r_2(p_1, p_2), \dots, r_n(p_{n-1}, p)$ in \mathcal{B}' . Thus, v, p_1, \dots, p_{n-1} must also be elements of the connected component, and these elements and the path must have pre-images in the kernel objects of $\exists X.\mathcal{A}$, i.e., there are kernel objects w_0, w_1, \dots, w_n in \mathcal{A} such that $h(w_0) = v = p_0, h(w_1) = p_1, \dots, h(w_{n-1}) = p_{n-1}, h(w_n) = p = p_n$ and the role assertions $r_1(w_0, w_1), r_2(w_1, w_2), \dots, r_n(w_{n-1}, w_n)$ belong to \mathcal{A} . We claim that this implies that n is smaller than the number of kernel objects of $\exists X.\mathcal{A}$. In fact, otherwise there are indices i, j such that $0 \leq i < j \leq n$ and $w_i = w_j$. This would imply $p_i = h(w_i) = h(w_j) = p_j$, which yields a contradiction since the paths p_i and p_j have a different length, and thus cannot be identical.

Second, consider a connected component that does not contain a kernel object of $\exists Y.\mathcal{B}$, which means that this component lies in the unfolded part of $\exists Y'.\mathcal{B}'$. Thus, viewed in a directed way, it is a tree. Let p be the shell path that is the root of this tree, and assume that m is its length and u is its target. Using the same argument as in the previous case, we can show that the length of any shell path occurring in the tree is bounded by m plus the number of kernel objects of $\exists X.\mathcal{A}$. Unfortunately, we do not have a bound for m . However, it is easy to see that the trees in the unfolded part of \mathcal{B}' starting with a path with target u are all isomorphic to the tree starting with p . In addition, if m is larger than the number of shell objects of $\exists Y.\mathcal{B}$, then there is also a shorter path with target u since then there must be a repetition of a shell object in p . Consequently, there is a path q with target u whose length is bounded by the number of shell objects of $\exists Y.\mathcal{B}$. Instead of the connected component whose root is p we can then use the corresponding subtree with root q in the image of the function. \square

We are now ready to show that testing for the existence of a homomorphism from $\exists X'.\mathcal{A}'$ to $\exists Y'.\mathcal{B}'$ can be done by an NP procedure. As a consequence of Lemma 18, we can guess the functional part of the simulation required by the characterisation in Lemma 17 in non-deterministic polynomial time. What remains then is to check whether the guessed function h can be extended to a simulation from $\exists X.\mathcal{A}$ to $\exists Y'.\mathcal{B}'$ that coincides with h on the kernel objects of $\exists X.\mathcal{A}$. The next lemma shows that this can be done in polynomial time. We denote that subset of \mathcal{A} that contains only kernel objects with \mathcal{A}^- .

Lemma 19. *Let f be a function from the kernel objects of $\exists X.\mathcal{A}$ to the objects of $\exists Y'.\mathcal{B}'$. Then there is a simulation \mathfrak{T} from $\exists X.\mathcal{A}$ to $\exists Y'.\mathcal{B}'$ whose restriction to the kernel objects of $\exists X.\mathcal{A}$ is equal to f iff the following two conditions are satisfied:*

1. f is a homomorphism from $\exists X.\mathcal{A}^-$ to $\exists Y'.\mathcal{B}'$;
2. there is a simulation \mathfrak{R} from $\exists X.\mathcal{A}$ to $\exists Y.\mathcal{B}$ such that the following holds for all kernel objects u of $\exists X.\mathcal{A}$: if $f(u)$ has target v , then $(u, v) \in \mathfrak{R}$.

Proof. For the only-if direction, the fact that f is the restriction of a simulation to the kernel objects of $\exists X.\mathcal{A}$ obviously implies that f is a homomorphism from $\exists X.\mathcal{A}^-$ to $\exists Y'.\mathcal{B}'$. To show Condition 2, we define \mathfrak{R} to be the composition of \mathfrak{T} with the simulation $\mathfrak{S}' = \{ (p, v) \mid v \text{ is the target of the shell path } p \text{ in } \mathcal{B}' \}$ from $\exists Y'.\mathcal{B}'$ to $\exists Y.\mathcal{B}$ (see Proposition 15, where we have written this simulation as a function). Assume that $f(u) = p$ for a kernel object u of \mathcal{A} , and that the target of p is v . Then $(u, p) \in \mathfrak{T}$ and $(p, v) \in \mathfrak{S}'$, which yields $(u, v) \in \mathfrak{R}$ as required.

To show the if direction, we define \mathfrak{T}' to be the composition of \mathfrak{R} with the simulation $\mathfrak{S} = \{ (v, p) \mid v \text{ is the target of the shell path } p \text{ in } \mathcal{B}' \}$ (see Proposition 15). Let u be a kernel object of $\exists X.\mathcal{A}$ and $p = f(u)$, and let v be the target of p . Then $(u, v) \in \mathfrak{R}$ (by Condition 2) and $(v, p) \in \mathfrak{S}$, which yields $(u, p) \in \mathfrak{T}'$. This shows that f , if viewed as a relation, is contained in \mathfrak{T}' . We obtain \mathfrak{T} from \mathfrak{T}' by removing all tuples (u, p) such that u is a kernel object of $\exists X.\mathcal{A}$ and $p \neq f(u)$.

It remains to show that \mathfrak{T} is a simulation. First, note that every $a \in \Sigma_1$ is a kernel object, and thus we have $f(a) = a$ by Condition 1. This yields $(a, a) \in \mathfrak{T}$ since f is contained in \mathfrak{T} . Second, assume that the object v in \mathcal{A} satisfies $A(v) \in \mathcal{A}$, and that $(v, p) \in \mathfrak{T}$. Then $(v, p) \in \mathfrak{T}'$ also holds, and since \mathfrak{T}' is a simulation, we obtain $A(p) \in \mathcal{B}'$. Third, assume that $r(u, v) \in \mathcal{A}$ and that $(u, p) \in \mathfrak{T}$. Since $\mathfrak{T} \subseteq \mathfrak{T}'$ and \mathfrak{T}' is a simulation, there is a shell path q in \mathcal{B}' such that $r(p, q) \in \mathcal{B}'$ and $(v, q) \in \mathfrak{T}'$. If $(v, q) \in \mathfrak{T}$, then we are done. Otherwise, v must be a kernel object such that $f(v) \neq q$. But then u also must be a kernel object (since kernel objects are not reachable from shell objects). Consequently, $r(u, v) \in \mathcal{A}^-$ and $p = f(u)$. Since f is a homomorphism from $\exists X.\mathcal{A}^-$ to $\exists Y'.\mathcal{B}'$, we obtain $r(p, f(v)) = r(f(u), f(v)) \in \mathcal{B}'$, which completes our proof that \mathfrak{T} is a simulation since $(v, f(v)) \in \mathfrak{T}$. \square

The three lemmas shown in this section provide us with an NP procedure for deciding the existence of a homomorphism between shell unfoldings.

Theorem 20. *Let $\exists X.\mathcal{A}$ and $\exists Y.\mathcal{B}$ be two finite qABoxes whose object sets are partitioned into kernel objects and shell objects as introduced at the beginning of this section, and let $\exists X'.\mathcal{A}'$ and $\exists Y'.\mathcal{B}'$ be their shell unfoldings. Then the problem of deciding whether there is a homomorphism from $\exists X'.\mathcal{A}'$ to $\exists Y'.\mathcal{B}'$ is NP-complete in the size of the input $\exists X.\mathcal{A}$ and $\exists Y.\mathcal{B}$.*

Proof. NP-hardness is trivial since the homomorphism existence problem for qABoxes is NP-complete, which corresponds to the special case of our problem where the input qABoxes $\exists X.\mathcal{A}$ and $\exists Y.\mathcal{B}$ contain no shell objects.

To decide whether there is a homomorphism from $\exists X'. \mathcal{A}'$ to $\exists Y'. \mathcal{B}'$, we first guess a function f from the kernel objects of $\exists X. \mathcal{A}$ to the objects of $\exists Y'. \mathcal{B}'$ such that the length of the shell paths $f(u)$ is bounded by $k + \ell$, where k is number of kernel objects of $\exists X. \mathcal{A}$ and ℓ is the number of shell objects of $\exists Y. \mathcal{B}$. The restriction to paths of this length is justified by Lemma 18. The function f has polynomial size and can thus be guessed in non-deterministic polynomial time. Given f , we can obviously check in polynomial time whether it satisfies Condition 1 of Lemma 19. To decide whether Condition 2 of this lemma is satisfied as well, it is sufficient to compute the largest simulation from $\exists X. \mathcal{A}$ to $\exists Y. \mathcal{B}$ (which can be done in polynomial time [30]), and check whether it contains all pairs (u, v) where v is the target of $f(u)$ for the kernel objects u of \mathcal{A} . \square

5.3 A Compactness Theorem for Shell Unfoldings

Recall from Section 5.1 that CQ-entailment between countable qABoxes need not enforce a homomorphism between them, although the converse direction holds. Even if the left-hand side is a shell unfolding no homomorphism must exist, as the following example shows.

Example 21. Let \mathbb{N} be the set of all natural numbers and \mathbb{Z} the set of all integers. From the qABox $\exists \{x\}. \{r(a, x), r(x, x)\}$ with kernel object a and shell object x we obtain as shell unfolding the *Ray* $\exists \{x_i \mid i \in \mathbb{N}\}. \{r(a, x_1), r(x_1, x_2), \dots\}$. Further consider the *Line* $\exists \{y_j \mid j \in \mathbb{Z}\}. \{\dots, r(y_{-2}, y_{-1}), r(y_{-1}, y_0), r(y_0, y_1), r(y_1, y_2), \dots\}$. While from every finite sub-qABox of the Line there is a homomorphism to the Ray, there is no homomorphism from the Line to the Ray.¹⁴

By viewing role assertions as edges, we can use the following notions from graph theory. A *path* in a qABox $\exists X. \mathcal{A}$ is a sequence $R_1(x_0, x_1), \dots, R_n(x_{n-1}, x_n)$ where each R_i is either a role name r_i or an inverse role r_i^- , and \mathcal{A} contains $r_i(x_{i-1}, x_i)$ if $R_i = r_i$, and \mathcal{A} contains $r_i(x_i, x_{i-1})$ if otherwise $R_i = r_i^-$. If the intermediate objects are irrelevant, we may write $x_0 \xrightarrow{R_1 \dots R_n} x_n$. If no object occurs twice, then the path is *simple*. The path is *directed* if all R_i are role names. Furthermore, this path is *upward* if the majority of the R_i are non-inverse role names, is *downward* if the majority of the R_i are inverse role names, and *sideward* otherwise. The *height* of this path is the number of non-inverse role names R_i minus the number of inverse role names R_i . An object u is called *lowest* if there is no downward path starting from u .

A *connected component* of $\exists X. \mathcal{A}$ is a subset U of $\text{Obj}(\exists X. \mathcal{A})$ such that each two objects in U are joined by a path and U contains all objects to which a path from some object in U exists. A *cycle* is a path in which the first and last object are equal. A *fork* consists of two paths $u \xrightarrow{R_1 \dots R_m} w$ and $v \xrightarrow{S_1 \dots S_n} w$. There are particular cycles and forks that any homomorphism to a shell unfolding must map to zero-length shell paths. A cycle is *forbidden* if it does not have the form $r_1(x_0, x_1), \dots, r_n(x_{n-1}, x_n), r_n^-(x_n, x'_n), \dots, r_1^-(x'_1, x_0)$, modulo shifting.

¹⁴ This example is a variant of [21, Example 5].

A fork is *forbidden* if it has the form $u \xrightarrow{r_1 \cdots r_n} w$ and $v \xrightarrow{s_1 \cdots s_n} w$ with $r_i \neq s_i$ for some i .

To prove the Compactness Theorem for shell unfoldings, we employ and adapt a Compactness Theorem for directed graphs [21–23], which builds upon Tychonoff’s Theorem [48], a very important result in general topology. The proof of the latter assumes the *Axiom of Choice* [49]: the product of non-empty sets is non-empty or, equivalently, for each set \mathcal{M} of non-empty sets, there is a *choice function* $f: \mathcal{M} \rightarrow \bigcup \mathcal{M}$ such that $f(M) \in M$ for each set $M \in \mathcal{M}$.

Proposition 22. *Assume the Axiom of Choice. Further let $\exists X'.\mathcal{A}'$ be the shell unfolding of a finite qABox $\exists X.\mathcal{A}$ and consider a countable qABox $\exists Y.\mathcal{B}$ in which every connected component contains an individual name, a forbidden cycle, a forbidden fork, or a lowest variable. The following statements are equivalent:*

1. $\exists X'.\mathcal{A}' \models_{\text{CQ}} \exists Y.\mathcal{B}$
2. For every finite sub-qABox of $\exists Y.\mathcal{B}$, there is a homomorphism to $\exists X'.\mathcal{A}'$.
3. There is a homomorphism from $\exists Y.\mathcal{B}$ to $\exists X'.\mathcal{A}'$.
4. $\exists X'.\mathcal{A}' \models \exists Y.\mathcal{B}$

Proof. 1 \Leftrightarrow 2. The only-if direction is trivial. Regarding the if direction, assume that $\exists Z.\mathcal{C}$ is a finite qABox entailed by $\exists Y.\mathcal{B}$. By Lemma 9 there is a homomorphism h from $\exists Z.\mathcal{C}$ to $\exists Y.\mathcal{B}$. The image of h is a finite sub-qABox of $\exists Y.\mathcal{B}$, from which there is a homomorphism k to $\exists X'.\mathcal{A}'$. The composition $k \circ h$ is a homomorphism from $\exists Z.\mathcal{C}$ to $\exists X'.\mathcal{A}'$. So, according to Lemma 9, $\exists Z.\mathcal{C}$ is entailed by $\exists X'.\mathcal{A}'$.

2 \Leftrightarrow 3. The if direction is trivial. We derive the only-if direction from [21, Lemma 16]. Although this lemma is formulated for directed graphs and the corresponding notion of a homomorphism, it similarly holds for qABoxes and their homomorphisms. The reason is that qABoxes are also directed graphs but additionally equipped with node and edge labels, and therefore the same proof argumentation applies to them.

So, we will build a mapping $\ell: \text{Obj}(\exists Y.\mathcal{B}) \rightarrow \wp(\text{Obj}(\exists X'.\mathcal{A}'))$ such that

- for each finite sub-qABox $\exists Z.\mathcal{C}$ of $\exists Y.\mathcal{B}$, there is a homomorphism from $\exists Z.\mathcal{C}$ to $\exists X'.\mathcal{A}'$ that sends each object u of $\exists Z.\mathcal{C}$ to an object in $\ell(u)$,
- $\ell(u)$ is finite for each $u \in \text{Obj}(\exists Y.\mathcal{B})$.

The first statement is exactly the first requirement on ℓ in [21, Lemma 16]. For the second statement, we can endow each $\ell(u)$ with the discrete topology to get a compact topology. Furthermore, then all subsets are closed in this topology, as are all subsets in the product topology, i.e., the second condition on ℓ in [21, Lemma 16] is trivially fulfilled. We can then conclude that there is a homomorphism from $\exists Y.\mathcal{B}$ to $\exists X'.\mathcal{A}'$.

First note that, although $\exists Y.\mathcal{B}$ is countably infinite if not finite, only finitely many names from the signature can occur in it. The reason is that $\exists Y.\mathcal{B}$ is CQ-entailed by the shell unfolding $\exists X'.\mathcal{A}'$, which is built from the finite qABox $\exists X.\mathcal{A}$.

- For each concept name A not occurring in $\exists X.\mathcal{A}$, the qABox $\exists\{x\}.\{A(x)\}$ is not entailed by $\exists X'.\mathcal{A}'$ and thus neither by $\exists Y.\mathcal{B}$, i.e., also in $\exists Y.\mathcal{B}$ the concept name A does not occur.
- This similarly holds for role names r , using the qABox $\exists\{x, y\}.\{r(x, y)\}$.
- The same follows for individual names a , but we need more qABoxes to verify this, namely $\exists\emptyset.\{A(a)\}$ for each $A \in \Sigma_I$, and $\exists\{x\}.\{r(x, a)\}$ as well as $\exists\{y\}.\{r(a, y)\}$ for each $r \in \Sigma_R$.

In the following, we will denote by \mathfrak{P}_i the set consisting of all shell paths with a length not exceeding i . Each \mathfrak{P}_i is finite since there are only finitely many kernel and shell objects from which the shell paths are built.

Since homomorphisms send each individual name a to itself, it only makes sense to define $\ell(a) := \{a\}$. Now consider a connected component U of $\exists Y.\mathcal{B}$. We distinguish three cases.

- (a) Assume that U contains an individual name, say a_U . Further let y be a variable in U . There is a shortest path from a_U to y . Every homomorphism h maps this path to a path from a_U to $h(y)$ in $\exists X'.\mathcal{A}'$, which need not be shortest though. It follows that $d(a_U, h(y)) \leq d(a_U, y)$, where $d(u, v)$ denotes the length of a shortest path from u to v in the respective qABox and $d(u, v) = \infty$ if there is no path from u to v .

Thus we define $\ell(y)$ as the set of all shell paths p for which the distance to a_U (in $\exists X'.\mathcal{A}'$) does not exceed the distance from a_U to y (in $\exists Y.\mathcal{B}$), i.e., $\ell(y) := \{p \mid p \in \text{Obj}(\exists X'.\mathcal{A}') \text{ and } d(a_U, p) \leq d(a_U, y)\}$.¹⁵ Since a_U is a zero-length shell path, we have $\ell(y) \subseteq \mathfrak{P}_{d(a_U, y)}$ and so $\ell(y)$ is finite.

We conclude that, if $\exists Z.\mathcal{C}$ is a finite sub-qABox of $\exists Y.\mathcal{B}$ that contains, for each variable $y \in U \cap Z$, one path from a_U to y that is shortest in $\exists Y.\mathcal{B}$, and if h is a homomorphism from $\exists Z.\mathcal{C}$ to the shell unfolding, then h is bounded by ℓ on U , i.e., $h(y) \in \ell(y)$ for each $y \in U \cap Z$.

- (b) Now assume that U contains a forbidden cycle or fork but no individual name. Choose one forbidden cycle or fork and denote it by CF_U . Every homomorphism maps all variables in CF_U to zero-length shell paths (i.e., kernel objects) since other shell paths are not on forbidden cycles or forks. Therefore, we choose $\ell(y) := \mathfrak{P}_0$ for each variable y in CF_U . We can further argue similar to the previous case. For every other variable y in U , there must be a shortest path from CF_U to y . A homomorphism h maps it to a path from some zero-length shell path to $h(y)$ in $\exists X'.\mathcal{A}'$, which need not be shortest though. Thus, $h(y)$ must be a shell path with a length not exceeding $d(CF_U, y)$ and we therefore set $\ell(y) := \mathfrak{P}_{d(CF_U, y)}$.

It follows that, if $\exists Z.\mathcal{C}$ is a finite sub-qABox of $\exists Y.\mathcal{B}$ that contains, for each variable $y \in U \cap Z$, one path from CF_U to y that is shortest in $\exists Y.\mathcal{B}$, and if h is a homomorphism from $\exists Z.\mathcal{C}$ to the shell unfolding, then h is bounded by ℓ on U , i.e., $h(y) \in \ell(y)$ for each $y \in U \cap Z$.

¹⁵ This is similar to the proof of [21, Theorem 18].

- (c) It remains the case where U contains no individuals and no forbidden cycles or forks. By assumption, U contains a lowest variable, say y_U . Recall that each shell path starts with a kernel object and otherwise consists only of shell objects, and denote by n the number of shell objects in $\exists X.\mathcal{A}$. Now, consider a homomorphism h from a finite sub-qABox of $\exists Y.\mathcal{B}$ to the shell unfolding $\exists X'.\mathcal{A}'$. If $h(y_U)$ has a length exceeding n , then by the pigeonhole principle $h(y_U)$ contains some shell object twice. In this case, by exhaustively cutting out non-empty sub-paths that start and end with the same shell object, we obtain a shorter shell path with the same target and with a length not exceeding n . Thus, it suffices to take $\ell(y_U) := \mathfrak{P}_n$.

Now consider another variable y in U . Since U is connected, there is a path from y_U to y . Since y_U is lowest, this path has a non-negative height, say j , and specifically cannot start with an inverse role name. Each homomorphism h maps it to a path between $h(y_U)$ and $h(y)$ in $\exists X'.\mathcal{A}'$ with same height j and upward first edge. Due to the structure of the shell unfolding and since $j \geq 0$, the shell path $h(y_U)$ is a prefix of $h(y)$. If we did not shorten $h(y_U)$, then $h(y)$ is a shell path of length not exceeding $n+j$ and so there is no need to shorten $h(y)$ either. Otherwise, we shorten $h(y)$ by replacing the prefix $h(y_U)$ with the shortening from above and thereby obtain a shell path with length bounded by $n+j$. We thus define $\ell(y) := \mathfrak{P}_{n+j}$.

Since for any homomorphism h from a finite sub-qABox of $\exists Y.\mathcal{B}$ to the shell unfolding, we cut out the same infix from all values $h(y)$ where y occurs in this sub-qABox, the mapping with the shortened values is still a homomorphism but now bounded by ℓ .

Consequently, if $\exists Z.\mathcal{C}$ is a finite sub-qABox of $\exists Y.\mathcal{B}$ that contains, for every variable $y \in U \cap Z$, one path from y_U to y in $\exists Y.\mathcal{B}$, and if h is a homomorphism from $\exists Z.\mathcal{C}$ to the shell unfolding, then we can construct from h a homomorphism k that is bounded by ℓ on U , i.e., $k(y) \in \ell(y)$ for each $y \in U \cap Z$.

Finally, consider a finite sub-qABox $\exists Z.\mathcal{C}$ of $\exists Y.\mathcal{B}$, and assume that the following closure properties are satisfied for each connected component U of $\exists Y.\mathcal{B}$:

- (C1) If U contains an individual name, and if y is a variable in $U \cap Z$, then $\exists Z.\mathcal{C}$ contains one path from a_U to y that is shortest in $\exists Y.\mathcal{B}$.
- (C2) If U contains a forbidden cycle or fork but no individual name, and if y is a variable in $U \cap Z$, then $\exists Z.\mathcal{C}$ contains one path from CF_U to y that is shortest in $\exists Y.\mathcal{B}$.
- (C3) If U contains no individual names and no forbidden cycles or forks, and if y is a variable in $U \cap Z$, then $\exists Z.\mathcal{C}$ contains one path from y_U to y in $\exists Y.\mathcal{B}$.

By assumption, there is a homomorphism h from $\exists Z.\mathcal{C}$ to $\exists X'.\mathcal{A}'$. As we have seen above, there is a such homomorphism k that sends each object u to an object in $\ell(u)$.

For other finite sub-qABoxes $\exists Z.C$ that do not satisfy the Conditions (C1), (C2), and (C3), we construct a larger finite sub-qABox that does. We simply add, for each $y \in Z$, one path from a_U resp. CF_U resp. y_U to y . This adds only finitely many objects and assertions. It is easy to see that afterwards the three closure conditions are fulfilled. As shown above, there is a homomorphism from this larger sub-qABox to $\exists X'.\mathcal{A}'$ that is bounded by ℓ . The restriction of this homomorphism to $\exists Z.C$ is, of course, a homomorphism from $\exists Z.C$ to $\exists X'.\mathcal{A}'$ and is bounded by ℓ .

3 \Leftrightarrow 4. Both directions follow from Lemma 9. \square

Since every shell unfolding is countable and every connected component contains an individual name, a forbidden cycle, a forbidden fork, or a lowest variable, Proposition 22 holds specifically when the qABox $\exists Y.\mathcal{B}$ on the right-hand side is a shell unfolding. Furthermore, since every finite qABox is a shell unfolding, the left-hand side $\exists X'.\mathcal{A}'$, the right-hand side $\exists Y.\mathcal{B}$, or both can also be finite qABoxes.

Lemma 23. *Every qABox of any cardinality is CQ-equivalent to a countable qABox in which every connected component contains an individual name, a forbidden cycle, a forbidden fork, or a lowest variable.*

Proof. Consider a qABox $\exists X.\mathcal{A}$. We obtain a CQ-equivalent qABox as the union of all finite qABoxes entailed by it:

$$\exists X.\mathcal{A} \equiv_{\text{CQ}} \bigcup \{ \exists Y.\mathcal{B} \mid \exists Y.\mathcal{B} \text{ is a finite qABox and } \exists X.\mathcal{A} \models \exists Y.\mathcal{B} \}.$$

Since the set of finite qABoxes is countable, the right-hand side is a countable union of finite qABoxes, hence a countable qABox. Further consider a connected component U without individual names, forbidden cycles, and forbidden forks. If there were no lowest variable in U , then U would be infinite, a contradiction. \square

Lemma 24. *Let $\exists X.\mathcal{A}$ and $\exists Y.\mathcal{B}$ be qABoxes of any cardinality, and further let \mathcal{T} be an \mathcal{EL} TBox. If $\exists X.\mathcal{A} \models_{\text{CQ}} \exists Y.\mathcal{B}$, then $\exists X.\mathcal{A} \models_{\text{CQ}}^{\mathcal{T}} \exists Y.\mathcal{B}$.*

Proof. Assume $\exists X.\mathcal{A} \models_{\text{CQ}} \exists Y.\mathcal{B}$. Further consider a finite qABox $\exists Z.C$ with $\exists Y.\mathcal{B} \models^{\mathcal{T}} \exists Z.C$. We can drop the quantifier on the left, and it holds that $\mathcal{B} \models^{\mathcal{T}} \exists Z.C$. By the Compactness Theorem for first-order logic, there is a finite subset \mathcal{B}_0 of the matrix \mathcal{B} such that $\mathcal{B}_0 \models^{\mathcal{T}} \exists Z.C$. Let Y_0 be the finite set of all variables occurring in \mathcal{B}_0 . We then have $\exists Y.\mathcal{B} \models \exists Y_0.\mathcal{B}_0$ and $\exists Y_0.\mathcal{B}_0 \models^{\mathcal{T}} \exists Z.C$. The former implies $\exists X.\mathcal{A} \models \exists Y_0.\mathcal{B}_0$, and then the latter yields $\exists X.\mathcal{A} \models^{\mathcal{T}} \exists Z.C$. \square

5.4 CQ-Saturations

Given a qABox $\exists X.\mathcal{A}$ and an \mathcal{EL} TBox \mathcal{T} , we consider the shell unfolding of the IQ-saturation $\text{sat}_{\text{IQ}}^{\mathcal{T}}(\exists X.\mathcal{A})$, where all objects of the sub-qABox $\exists X.\mathcal{A}$ are kernel objects and all other objects (added by applications of the IQ-Saturation Rule)

are shell objects. We will show that it entails exactly those qABoxes that are CQ-entailed by $\exists X.A$ and \mathcal{T} . It can thus replace the finite CQ-saturation from [9] but is not limited to cycle-restricted TBoxes. For this reason, we denote this shell unfolding by $\text{sat}_{\text{CQ}}^{\mathcal{T}}(\exists X.A)$ and call it the CQ-saturation of $\exists X.A$ w.r.t. \mathcal{T} .

Proposition 25. *Let $\exists X.A$ be a finite qABox and \mathcal{T} an \mathcal{EL} TBox. For each qABox $\exists Z.C$ of any cardinality, $\exists X.A \models_{\text{CQ}}^{\mathcal{T}} \exists Z.C$ iff $\text{sat}_{\text{CQ}}^{\mathcal{T}}(\exists X.A) \models_{\text{CQ}} \exists Z.C$.*

Proof. According to Lemma 23, there is a countable qABox $\exists \bar{Z}.\bar{C}$ that satisfies the conditions in Proposition 22 and is CQ-equivalent to $\exists Z.C$, i.e., we have $\exists Z.C \equiv_{\text{CQ}} \exists \bar{Z}.\bar{C}$. Lemma 24 yields $\exists Z.C \equiv_{\text{CQ}}^{\mathcal{T}} \exists \bar{Z}.\bar{C}$. To prove the claim it thus suffices to verify that $\exists X.A \models_{\text{CQ}}^{\mathcal{T}} \exists \bar{Z}.\bar{C}$ iff $\text{sat}_{\text{CQ}}^{\mathcal{T}}(\exists X.A) \models_{\text{CQ}} \exists \bar{Z}.\bar{C}$. In the following, let $\exists Y.B$ be the IQ-saturation $\text{sat}_{\text{IQ}}^{\mathcal{T}}(\exists X.A)$ and let $\exists Y'.B'$ be its shell unfolding, which is $\text{sat}_{\text{CQ}}^{\mathcal{T}}(\exists X.A)$.

We start with the only-if direction and therefore assume that $\exists X.A \models_{\text{CQ}}^{\mathcal{T}} \exists \bar{Z}.\bar{C}$. We will verify that the canonical model $\text{CMod}(\exists Y'.B')$, which we denote by \mathcal{I} , is a model of the qABox $\exists X.A$ and of the TBox \mathcal{T} . We then conclude that every finite qABox entailed by $\exists \bar{Z}.\bar{C}$ w.r.t. \mathcal{T} is satisfied in \mathcal{I} . By Lemma 9, $\exists Y'.B' \models_{\text{CQ}} \exists \bar{Z}.\bar{C}$.

- Recall that $\exists X.A$ is a sub-qABox of the IQ-saturation $\exists Y.B$ and that the objects of $\exists X.A$ are the kernel objects of $\exists Y.B$. According to the definition of the shell unfolding $\exists Y'.B'$, all assertions involving kernel objects are copied over from $\exists Y.B$ to $\exists Y'.B'$, and thus $\exists X.A$ is also a sub-ABox of $\exists Y'.B'$. According to Lemma 9, the canonical model \mathcal{I} is a model of the shell unfolding $\exists Y'.B'$, from which we conclude that \mathcal{I} is a model of $\exists X.A$.
- Next, we prove that \mathcal{I} is a model of \mathcal{T} . The relation \mathfrak{S} in Proposition 15 is a simulation from the IQ-saturation $\exists Y.B$ to the shell unfolding $\exists Y'.B'$, and its inverse \mathfrak{S}^{-} is a simulation in the converse direction. Now, let $p \in \text{Dom}(\mathcal{I})$ and $C \sqsubseteq D \in \mathcal{T}$ such that $p \in C^{\mathcal{I}}$. It follows from Lemma 9 that the matrix B' entails $C(p)$, namely when we drop the quantifier of the qABox on the left and use $C(p)$ as qABox on the right. Due to the simulation \mathfrak{S}^{-} we obtain with Lemma 3 that B entails $C(u)$ where u is the target of p . Since $\exists Y.B$ is the IQ-saturation, the IQ-Saturation Rule is not applicable to it and thus B also entails $D(u)$. Applying Lemma 3 for the simulation \mathfrak{S} yields that B' entails $D(p)$. Again with Lemma 9 we finally obtain that $p \in D^{\mathcal{I}}$.

Regarding the if direction, assume $\exists Y'.B' \models_{\text{CQ}} \exists \bar{Z}.\bar{C}$. Proposition 22 yields $\exists Y'.B' \models \exists \bar{Z}.\bar{C}$. We will show that the latter implies $\exists X.A \models_{\text{CQ}}^{\mathcal{T}} \exists \bar{Z}.\bar{C}$, from which we conclude that $\exists X.A \models_{\text{CQ}}^{\mathcal{T}} \exists \bar{Z}.\bar{C}$.

Thus, assume that \mathcal{I} is a model of $\exists X.A$ and \mathcal{T} . In the proof of Proposition IV of [10] it is shown how one can construct a simulation \mathfrak{S}_n from $\exists Y.B$ to \mathcal{I} whose restriction \mathfrak{S}_0 to the objects of $\exists X.A$, which are the kernel objects of $\exists Y.B$, is functional. Using an adaptation of Lemma 17 to a setting where one considers the corresponding characterization of the existence of a homomorphism from a shell unfolding into an interpretation, we obtain that there is a homomorphism from $\exists Y'.B'$ into \mathcal{I} , i.e., \mathcal{I} is a model of $\exists Y'.B'$ and thus also of $\exists \bar{Z}.\bar{C}$. \square

Example 26. Coming back to Example 1, where we constructed the IQ-saturation with kernel object n and shell object x_V , we now obtain as shell unfolding the CQ-saturation $\text{sat}_{\text{CQ}}^{\mathcal{T}}(\exists X.\mathcal{A}) = \exists \{x_1, x_2, \dots\}. \{V(n), \ell(n, x_1), V(x_1), \ell(x_1, x_2), V(x_2), \dots\}$, where $x_k := n \xrightarrow{\ell} x_V \xrightarrow{\ell} \dots \xrightarrow{\ell} x_V$.

$$\text{sat}_{\text{CQ}}^{\mathcal{T}}(\exists X.\mathcal{A}): \quad \begin{array}{ccccccc} & V & & V & & V & & V & & \\ & \circ & & \circ & & \circ & & \circ & & \\ \xrightarrow{\ell} & & \xrightarrow{\ell} & & \xrightarrow{\ell} & & \xrightarrow{\ell} & & \xrightarrow{\ell} & \dots \end{array}$$

5.5 Proof of Proposition 2 for arbitrary TBoxes

The if direction of Proposition 2 has been proven for arbitrary TBoxes, but the only-if direction only for cycle-restricted TBoxes. Now that we know how saturations of qABoxes can be constructed also w.r.t. TBoxes that need not be cycle-restricted, we can prove the only-if direction for all TBoxes. We do not need to change much, and only need to modify how the first qABox $\exists Z_0.\mathcal{C}_0$ in the sequence is constructed.

Proof. Assume that $\exists X.\mathcal{A} \models^{\mathcal{T}} \exists Y.\mathcal{B}$, i.e., there is a homomorphism h from $\exists Y.\mathcal{B}$ to $\text{sat}_{\text{CQ}}^{\mathcal{T}}(\exists X.\mathcal{A})$, cf. Lemma 9 and Proposition 25. Since $\exists Y.\mathcal{B}$ is finite, the image $h(\mathcal{B}) := \{A(h(u)) \mid A(u) \in \mathcal{B}\} \cup \{r(h(u), h(v)) \mid r(u, v) \in \mathcal{B}\}$ is a finite subset of the matrix of $\text{sat}_{\text{CQ}}^{\mathcal{T}}(\exists X.\mathcal{A})$. Assume that P_h is the set of all shell paths occurring in $h(\mathcal{B})$. Then, we obtain a finite sub-qABox $\exists Z.\mathcal{C}$ of $\text{sat}_{\text{CQ}}^{\mathcal{T}}(\exists X.\mathcal{A})$ as follows: its object set consists of all kernel objects (i.e., all objects of $\exists X.\mathcal{A}$) and of all shell paths that are itself in P_h or are a prefix of a path in P_h , and its matrix \mathcal{C} consists of all assertions in $\text{sat}_{\text{CQ}}^{\mathcal{T}}(\exists X.\mathcal{A})$ involving only these objects.

The finite qABox $\exists Z.\mathcal{C}$ can be constructed from $\exists X.\mathcal{A}$ by means of the CQ-Saturation Rule. Since $h(\mathcal{B})$ is a subset of \mathcal{C} , it follows that h is already a homomorphism from $\exists Y.\mathcal{B}$ to $\exists Z.\mathcal{C}$. Thus, we take $\exists Z_0.\mathcal{C}_0 := \exists Z.\mathcal{C}$ and $h_0 := h$. The remainder of the proof of the only-if direction is as in Section 2.3. \square

5.6 CQ-Repairs

Recall that the finite CQ-repairs in [9] can only be constructed w.r.t. cycle-restricted TBoxes. We will now drop this restriction, with the effect that CQ-repairs could become infinite. However, we show that they can still be finitely described, namely as shell unfoldings of canonical IQ-repairs. Concerning the shell unfolding of such an IQ-repair, an object $\langle u, \mathcal{K} \rangle$ is a kernel object if u is a kernel object (in the underlying IQ-saturation), and otherwise it is a shell object.

Lemma 27. *For each repair seed \mathcal{S} , the shell unfolding of its induced IQ-repair $\text{rep}_{\text{IQ}}^{\mathcal{T}}(\exists X.\mathcal{A}, \mathcal{S})$ is a CQ-repair. We thus denote it by $\text{rep}_{\text{CQ}}^{\mathcal{T}}(\exists X.\mathcal{A}, \mathcal{S})$ and call it the canonical CQ-repair induced by \mathcal{S} .*

Proof. We first show that $\exists X.\mathcal{A} \models_{\text{CQ}}^{\mathcal{T}} \text{rep}_{\text{CQ}}^{\mathcal{T}}(\exists X.\mathcal{A}, \mathcal{S})$. According to Propositions 22 and 25 it suffices to show that there is a homomorphism from $\text{rep}_{\text{CQ}}^{\mathcal{T}}(\exists X.\mathcal{A}, \mathcal{S})$ to $\text{sat}_{\text{CQ}}^{\mathcal{T}}(\exists X.\mathcal{A})$. Recall that $\text{sat}_{\text{CQ}}^{\mathcal{T}}(\exists X.\mathcal{A})$ is the shell unfolding $\exists Y'.\mathcal{B}'$ of the IQ-saturation $\exists Y.\mathcal{B}$ of $\exists X.\mathcal{A}$. In addition, $\text{rep}_{\text{CQ}}^{\mathcal{T}}(\exists X.\mathcal{A}, \mathcal{S})$ is the shell unfolding $\exists Z'.\mathcal{C}'$ of the canonical IQ-repair $\exists Z.\mathcal{C} := \text{rep}_{\text{IQ}}^{\mathcal{T}}(\exists X.\mathcal{A}, \mathcal{S})$. Thus, we will show that there is a homomorphism from $\exists Z'.\mathcal{C}'$ to $\exists Y'.\mathcal{B}'$. The proof of Proposition 8 in [10] shows that there is a homomorphism from $\exists Z.\mathcal{C}$ to $\exists Y.\mathcal{B}$ that maps the kernel objects of $\exists Z.\mathcal{C}$ to kernel objects of $\exists Y.\mathcal{B}$. In addition, by Proposition 15, there is a simulation from $\exists Y.\mathcal{B}$ to $\exists Y'.\mathcal{B}'$ that is the identity on the kernel objects of $\exists Y.\mathcal{B}$. By composing the former homomorphism with the latter simulation, we obtain a simulation from $\exists Z.\mathcal{C}$ to $\exists Y'.\mathcal{B}'$ that is functional on the kernel objects of $\exists Z.\mathcal{C}$. Lemma 17 implies that there is a homomorphism from $\exists Z'.\mathcal{C}'$ to $\exists Y'.\mathcal{B}'$.

It remains to show that $\text{rep}_{\text{CQ}}^{\mathcal{T}}(\exists X.\mathcal{A}, \mathcal{S})$ does not entail any assertion $C(a) \in \mathcal{P}$. By Proposition 15, $\text{rep}_{\text{IQ}}^{\mathcal{T}}(\exists X.\mathcal{A}, \mathcal{S}) = \exists Z.\mathcal{C}$ CQ-entails $\text{rep}_{\text{CQ}}^{\mathcal{T}}(\exists X.\mathcal{A}, \mathcal{S}) = \exists Z'.\mathcal{C}'$. Thus, if $\text{rep}_{\text{CQ}}^{\mathcal{T}}(\exists X.\mathcal{A}, \mathcal{S})$ entailed $C(a) \in \mathcal{P}$, then this assertion would also be entailed by $\text{rep}_{\text{IQ}}^{\mathcal{T}}(\exists X.\mathcal{A}, \mathcal{S})$. Since we know that $\text{rep}_{\text{IQ}}^{\mathcal{T}}(\exists X.\mathcal{A}, \mathcal{S})$ is an IQ-repair, this cannot be the case. \square

Lemma 28. *Every canonical CQ-repair $\text{rep}_{\text{CQ}}^{\mathcal{T}}(\exists X.\mathcal{A}, \mathcal{S})$ is saturated w.r.t. \mathcal{T} .*

Proof. We denote $\text{rep}_{\text{CQ}}^{\mathcal{T}}(\exists X.\mathcal{A}, \mathcal{S})$ by $\exists Y'.\mathcal{B}'$. According to Definition 10, we need to show that for each CI $C \sqsubseteq D$ in \mathcal{T} and for each object u of $\exists Y'.\mathcal{B}'$, if $\mathcal{B}' \models C(u)$, then $\mathcal{B}' \models D(u)$. This is the case iff the Saturation Rule is not applicable to any object. We already know that the canonical IQ-repair for the same repair seed \mathcal{S} is saturated [9], and we denote it by $\exists Y.\mathcal{B}$. We verify that its shell unfoldings, which is the considered CQ-repair, is saturated as well. Therefore, let u be an object of $\exists Y'.\mathcal{B}'$ with $\mathcal{B}' \models C(u)$ for some CI $C \sqsubseteq D$ in \mathcal{T} . The homomorphism h from Proposition 15 sends u to an object $h(u)$ with $\mathcal{B} \models C(h(u))$. Since $\exists Y.\mathcal{B}$ is saturated, it follows that $\mathcal{B} \models D(h(u))$. Thus, with the simulation \mathfrak{S} from Proposition 15 we obtain that $\mathcal{B}' \models D(u)$ as $(h(u), u) \in \mathfrak{S}$. \square

Next, we show that the canonical CQ-repairs are complete in the sense that every countable CQ-repair is entailed by a canonical one.

Lemma 29. *If $\exists W.\mathcal{D}$ is a CQ-repair of $\exists X.\mathcal{A}$ for \mathcal{P} w.r.t. \mathcal{T} of any cardinality, then there is a repair seed \mathcal{S} such that $\text{rep}_{\text{CQ}}^{\mathcal{T}}(\exists X.\mathcal{A}, \mathcal{S}) \models_{\text{CQ}}^{\mathcal{T}} \exists W.\mathcal{D}$.*

Proof. According to Lemma 23, there is a countable qABox $\exists \bar{W}.\bar{\mathcal{D}}$ that satisfies the conditions in Proposition 22 and is CQ-equivalent to $\exists W.\mathcal{D}$, i.e., we have $\exists W.\mathcal{D} \equiv_{\text{CQ}} \exists \bar{W}.\bar{\mathcal{D}}$. Lemma 24 yields $\exists W.\mathcal{D} \equiv_{\text{CQ}}^{\mathcal{T}} \exists \bar{W}.\bar{\mathcal{D}}$. To prove the claim it thus suffices to verify that there is a repair seed \mathcal{S} such that $\text{rep}_{\text{CQ}}^{\mathcal{T}}(\exists X.\mathcal{A}, \mathcal{S}) \models_{\text{CQ}}^{\mathcal{T}} \exists \bar{W}.\bar{\mathcal{D}}$.

This lemma generalizes Proposition 8 in [10]. Its proof is actually very similar, with one exception that will be explained below. Let $\exists Y.\mathcal{B}$ be the IQ-saturation of the input qABox $\exists X.\mathcal{A}$, from which we obtain as shell unfolding the CQ-saturation, denoted by $\exists Y'.\mathcal{B}'$. As a CQ-repair, $\exists \bar{W}.\bar{\mathcal{D}}$ is CQ-entailed by $\exists X.\mathcal{A}$

w.r.t. \mathcal{T} and thus Propositions 22 and 25 yields a homomorphism h from $\exists \bar{W}. \bar{\mathcal{D}}$ to $\exists Y'. \mathcal{B}'$.

As first step, we define the function $\mathcal{F}: \text{Obj}(\exists \bar{W}. \bar{\mathcal{D}}) \rightarrow \wp(\text{Atoms}(\mathcal{T}, \mathcal{P}))$ as follows: for each object t of $\exists \bar{W}. \bar{\mathcal{D}}$, let u be the target of $h(t)$ and define

$$\mathcal{F}(t) := \text{Max}_{\sqsubseteq \emptyset} \{ C \mid C \in \text{Atoms}(\mathcal{T}, \mathcal{P}), \mathcal{B} \models C(u), \bar{\mathcal{D}} \not\models^{\mathcal{T}} C(t) \}.$$

Like in the proof of Proposition 8 in [10], each $\mathcal{F}(t)$ is a repair type for u .

From \mathcal{F} we obtain the repair seed \mathcal{S} with $\mathcal{S}_a := \mathcal{F}(a)$ for each individual name a . The IQ-repair induced by \mathcal{S} is denoted by $\exists Z. \mathcal{C}$, and as its shell unfolding we obtain the CQ-repair which we denote by $\exists Z'. \mathcal{C}'$.

Next, we will define a mapping k from objects of $\exists \bar{W}. \bar{\mathcal{D}}$ to objects of $\exists Z'. \mathcal{C}'$. In the proof of Proposition 8 in [10] we defined $k(t) := \langle\langle h(t), \mathcal{F}(t) \rangle\rangle$. Since the CQ-saturation $\exists Y'. \mathcal{B}'$ and the CQ-repair $\exists Z'. \mathcal{C}'$ are both shell unfoldings, h now maps to shell paths and k also must map to shell paths, which is why we cannot define k as before. In fact, we define it so that the target of $k(t)$ equals $\langle\langle u, \mathcal{F}(t) \rangle\rangle$ where u is the target of $h(t)$.

To this end, consider an object $t \in \text{Obj}(\exists \bar{W}. \bar{\mathcal{D}})$, assume $h(t) = t_0 \xrightarrow{r_1} t_1 \xrightarrow{r_2} \dots \xrightarrow{r_n} t_n$, and define the value $k(t)$ according to the following case distinction:

- If $n = 0$ (i.e., if $h(t)$ is a kernel object), then define

$$k(t) := \langle\langle t_0, \mathcal{F}(t) \rangle\rangle.$$

- Otherwise, if t has no r_n -predecessor, then define

$$k(t) := \langle\langle t_0, \emptyset \rangle\rangle \xrightarrow{r_1} \langle\langle t_1, \emptyset \rangle\rangle \xrightarrow{r_2} \dots \xrightarrow{r_{n-1}} \langle\langle t_{n-1}, \emptyset \rangle\rangle \xrightarrow{r_n} \langle\langle t_n, \mathcal{F}(t) \rangle\rangle.$$

Remark. Since $h(t) = t_0 \xrightarrow{r_1} t_1 \xrightarrow{r_2} \dots \xrightarrow{r_n} t_n$ is a shell path in the IQ-saturation $\exists Y. \mathcal{B}$, it must contain the role assertions $r_1(t_0, t_1), \dots, r_n(t_{n-1}, t_n)$. Furthermore, \emptyset is a repair type for all objects of $\exists Y. \mathcal{B}$, and thus each object in the path $k(t)$ is really an object of the IQ-repair. That these objects are connected by role assertions is vacuously true since the empty repair types do not contain any existential restrictions and thus Condition (CR2) is satisfied. Since t_0 is a kernel object and t_1, \dots, t_n are shell objects, $\langle\langle t_0, \emptyset \rangle\rangle$ is a kernel object and $\langle\langle t_1, \emptyset \rangle\rangle, \dots, \langle\langle t_{n-1}, \emptyset \rangle\rangle$ as well as $\langle\langle t_n, \mathcal{F}(t) \rangle\rangle$ are shell objects. We conclude that $k(t)$ is a shell path in the IQ-repair $\exists Z. \mathcal{C}$ and thus an object of the CQ-repair $\exists Z'. \mathcal{C}'$.

- Otherwise, let u be some r_n -predecessor of t and define

$$k(t) := k(u) \xrightarrow{r_n} \langle\langle t_n, \mathcal{F}(t) \rangle\rangle.$$

Remark. The predecessor u need not be unique, but $h(u) = t_0 \xrightarrow{r_1} t_1 \xrightarrow{r_2} \dots \xrightarrow{r_{n-1}} t_{n-1}$ is enforced since $r_n(u, t) \in \bar{\mathcal{D}}$ implies $r_n(h(u), h(t)) \in \mathcal{B}'$. We will see below that $r_n(u, t) \in \bar{\mathcal{D}}$ implies $r_n(\langle\langle u, \mathcal{F}(u) \rangle\rangle, \langle\langle t, \mathcal{F}(t) \rangle\rangle)$ and thus $k(t)$ is always a shell path.

We proceed by verifying that k is a homomorphism.

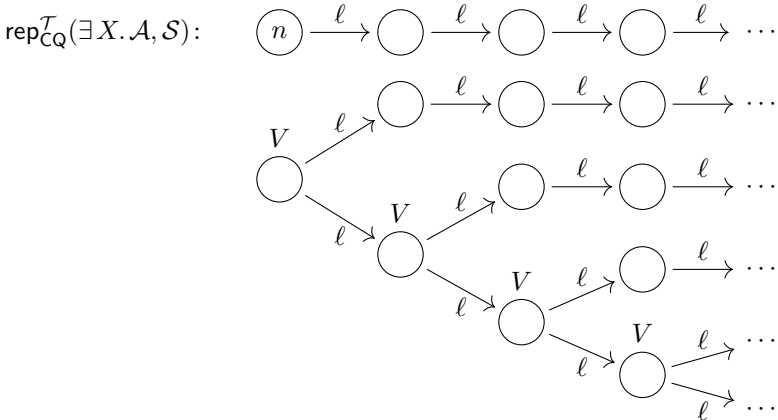
1. Let $A(t) \in \bar{\mathcal{D}}$. Then $A \notin \mathcal{F}(t)$ by definition of \mathcal{F} . Since h is a homomorphism, we infer $A(h(t)) \in \mathcal{B}'$ and thus $A(t_n) \in \mathcal{B}$. According to the definition of canonical IQ-repairs we obtain that $A(\langle\langle t_n, \mathcal{F}(t) \rangle\rangle) \in \mathcal{C}$, and thus $A(k(t)) \in \mathcal{C}'$.
2. Consider a role assertion $r(t, v) \in \bar{\mathcal{D}}$. By the homomorphism h we obtain $r(h(t), h(v)) \in \mathcal{B}'$, and so there must be a role assertion $r(t_n, t_{n+1}) \in \mathcal{B}$ where t_{n+1} is the target of the shell path $h(v)$.
 - If $h(v)$ is a kernel object, then $h(t)$ is also a kernel object, and we have $k(t) = \langle\langle t_n, \mathcal{F}(t) \rangle\rangle = \langle\langle h(t), \mathcal{F}(t) \rangle\rangle$ and $k(v) = \langle\langle t_{n+1}, \mathcal{F}(v) \rangle\rangle = \langle\langle h(v), \mathcal{F}(v) \rangle\rangle$.
 - If $h(v)$ is a shell path of length 1 or greater, then $k(t)$ ends with target $\langle\langle t_n, \mathcal{F}(t) \rangle\rangle$ and we have $k(v) = k(t) \xrightarrow{r} \langle\langle t_{n+1}, \mathcal{F}(v) \rangle\rangle$.

It remains to show that \mathcal{C}' contains the role assertion $r(k(t), k(v))$, for which we verify that \mathcal{C} contains the role assertion $r(\langle\langle t_n, \mathcal{F}(t) \rangle\rangle, \langle\langle t_{n+1}, \mathcal{F}(v) \rangle\rangle)$. We already know that $r(t_n, t_{n+1}) \in \mathcal{B}$, and it remains to show that $\text{Succ}(\mathcal{F}(t), r, t_{n+1}) \leq^{\emptyset} \mathcal{F}(v)$. This can be done in a similar way as in the proof of Proposition 8 in [10].

Finally, since k is a homomorphism from $\exists \bar{W}. \bar{\mathcal{D}}$ to $\exists Z'. \mathcal{C}'$, Proposition 22 yields that $\exists Z'. \mathcal{C}$ CQ-entails $\exists \bar{W}. \bar{\mathcal{D}}$, and thus also w.r.t. \mathcal{T} by Lemma 24. \square

The canonical CQ-repair must be constructed as shell unfolding of the full canonical IQ-repair, not from the optimized IQ-repair or another qABox that is IQ-equivalent. Otherwise, the completeness proof would not go through, as mapping targets for the constructed homomorphism k (see the above proof) might not be available. The next example illustrates this.

Example 30. In the canonical IQ-repair in Example 7 the individual n is the only kernel object, and the variables y_1, y_2, y_3 are the shell objects. As shell unfolding we obtain the following qABox.



For instance, the uppermost path consists of the shell paths $n, n \xrightarrow{\ell} y_2, n \xrightarrow{\ell} y_2 \xrightarrow{\ell} y_2, \dots$, and the lowermost path consists of the shell paths $y_1, y_1 \xrightarrow{\ell} y_3, y_1 \xrightarrow{\ell} y_3 \xrightarrow{\ell} y_3, \dots$. After reaching y_3 in a shell path we could also continue it with the role

assertion $\ell(y_3, y_2)$, after which the shell path could only be extended with the role assertion $\ell(y_2, y_2)$; these are all the shell paths in the middle.

If, instead, we would have constructed the shell unfolding from the optimized IQ-repair (see Page 20), then it would only contain the upper most path. Since it then does not contain any object that is an instance of V , the Boolean CQ $\exists x.V(x)$ would not be entailed, which indicates that knowledge is lost that should have been preserved in an optimal CQ-repair.

As an immediate consequence of the previous two lemmas we obtain the main result of this section.

Theorem 31. *Let $\exists X.A$ be a finite qABox, \mathcal{T} an \mathcal{EL} TBox, and \mathcal{P} a repair request. Then we can compute, in (deterministic) exponential time using an NP-oracle, a finite set of repair seeds $\{\mathcal{S}_1, \dots, \mathcal{S}_m\}$ such that the set $\{\text{rep}_{\text{CQ}}^{\mathcal{T}}(\exists X.A, \mathcal{S}_1), \dots, \text{rep}_{\text{CQ}}^{\mathcal{T}}(\exists X.A, \mathcal{S}_m)\}$ consists of all optimal CQ-repairs of $\exists X.A$ for \mathcal{P} w.r.t. \mathcal{T} (up to CQ-equivalence). The latter set is complete in the sense that every CQ-repair is CQ-entailed by an element of it.*

Proof. As shown in [9], there are exponentially many repair seeds in the size of \mathcal{P} and \mathcal{T} , which can be computed in exponential time. Not every canonical repair $\text{rep}_{\text{CQ}}^{\mathcal{T}}(\exists X.A, \mathcal{S})$ induced by a seed function \mathcal{S} is optimal, but we can check optimality of $\text{rep}_{\text{CQ}}^{\mathcal{T}}(\exists X.A, \mathcal{S})$ by testing whether there is no other seed function \mathcal{S}' such that $\text{rep}_{\text{CQ}}^{\mathcal{T}}(\exists X.A, \mathcal{S}')$ CQ-entails $\text{rep}_{\text{CQ}}^{\mathcal{T}}(\exists X.A, \mathcal{S})$, but not vice versa. This requires exponentially many entailment tests. To check whether $\text{rep}_{\text{CQ}}^{\mathcal{T}}(\exists X.A, \mathcal{S}') \models^{\mathcal{T}} \text{rep}_{\text{CQ}}^{\mathcal{T}}(\exists X.A, \mathcal{S})$, it is sufficient to decide whether there is a homomorphism from $\text{rep}_{\text{CQ}}^{\mathcal{T}}(\exists X.A, \mathcal{S})$ to $\text{rep}_{\text{CQ}}^{\mathcal{T}}(\exists X.A, \mathcal{S}')$, see Lemmas 11 and 28 and Proposition 22.

Since the canonical CQ-repairs are the shell-unfoldings of the corresponding canonical IQ-repairs, we can test for the existence of a homomorphism from $\text{rep}_{\text{CQ}}^{\mathcal{T}}(\exists X.A, \mathcal{S})$ to $\text{rep}_{\text{CQ}}^{\mathcal{T}}(\exists X.A, \mathcal{S}')$ by first computing $\text{rep}_{\text{IQ}}^{\mathcal{T}}(\exists X.A, \mathcal{S})$ and $\text{rep}_{\text{IQ}}^{\mathcal{T}}(\exists X.A, \mathcal{S}')$, which is possible in exponential time, and then applying as NP-oracle the NP-procedure testing for the existence of a homomorphism between their shell unfoldings (see Theorem 20). The computed set of optimal repairs is complete since the set of canonical CQ-repairs is complete according to Lemma 29. \square

Now assume that we are given a repair seed \mathcal{S} that induces an optimal CQ-repair $\text{rep}_{\text{CQ}}^{\mathcal{T}}(\exists X.A, \mathcal{S})$. If we want to work with this repair, we must be able to answer conjunctive queries on it. The following result shows that this has the same complexity as answering conjunctive queries on $\exists X.A$. Recall that a (Boolean) conjunctive query is just a finite qABox.

Proposition 32. *Let $\exists X.A$ and $\exists Z.C$ be (finite) qABoxes, \mathcal{T} an \mathcal{EL} TBox, \mathcal{P} a repair request, and \mathcal{S} a repair seed. We can decide if $\text{rep}_{\text{CQ}}^{\mathcal{T}}(\exists X.A, \mathcal{S}) \models^{\mathcal{T}} \exists Z.C$ in non-deterministic polynomial time w.r.t. the size of $\exists X.A$, \mathcal{T} , \mathcal{P} , and $\exists Z.C$.*

Proof. Basically, we want to adapt the approach used to prove Theorem 20 to test for the existence of a homomorphism from $\exists Z.C$ to $\text{rep}_{\text{CQ}}^{\mathcal{T}}(\exists X.A, \mathcal{S})$. This is

sufficient since $\text{rep}_{\text{CQ}}^{\mathcal{T}}(\exists X.\mathcal{A}, \mathcal{S})$ is saturated. Similar to the proof of Lemma 18, we look at the connected components of $\exists Z.C$.

If such a connected component contains an individual name, then we can show that elements of this connected component can only be mapped by a homomorphism to shell paths in $\text{rep}_{\text{CQ}}^{\mathcal{T}}(\exists X.\mathcal{A}, \mathcal{S})$ whose length is bounded by the number of objects of $\exists Z.C$. The objects of $\text{rep}_{\text{IQ}}^{\mathcal{T}}(\exists X.\mathcal{A}, \mathcal{S})$ are of the form $\langle\langle u, \mathcal{K} \rangle\rangle$ where u is an object of the IQ-saturation of $\exists X.\mathcal{A}$, whose size is polynomial in the size of $\exists X.\mathcal{A}$, and \mathcal{K} is a set of atoms occurring in \mathcal{P} or \mathcal{T} . Thus, guessing a polynomially long shell path consisting of such objects requires only polynomially many choices. This shows that we can guess a mapping from such a connected component into $\text{rep}_{\text{CQ}}^{\mathcal{T}}(\exists X.\mathcal{A}, \mathcal{S})$ in nondeterministic polynomial time. We can then test in polynomial time whether the guessed mapping is a homomorphism.

For a connected component not containing an individual name, we claim that it is sufficient to test whether $\exists X.\mathcal{A}$ entails this connected component w.r.t. \mathcal{T} . In fact, if the repair entails the connected component w.r.t. \mathcal{T} , then so does $\exists X.\mathcal{A}$ since every CQ-repair is CQ-entailed by $\exists X.\mathcal{A}$. Conversely, if $\exists X.\mathcal{A}$ entails the connected component q w.r.t. \mathcal{T} , then there is a homomorphism from q into the CQ-saturation of $\exists X.\mathcal{A}$. We claim that a copy of this CQ-saturation can be found in $\text{rep}_{\text{CQ}}^{\mathcal{T}}(\exists X.\mathcal{A}, \mathcal{S})$. This copy is obtained as the unfolding of the part of $\text{rep}_{\text{IQ}}^{\mathcal{T}}(\exists X.\mathcal{A}, \mathcal{S})$ consisting of the objects of the form $\langle\langle u, \emptyset \rangle\rangle$. Since q does not contain individual names, the homomorphism from q to $\exists X.\mathcal{A}$ can be turned into one from q into this copy. \square

6 Conclusion

In the first part of this paper we have mainly recalled the approaches and results from [9, 19]. In other work, we have extended these results in several directions. The paper [13] extends the expressivity of the underlying DL considerably, by adding nominals, inverse roles, regular role inclusions and the bottom concept to \mathcal{EL} , which yields a fragment of the well-known DL Horn-*SR*OIQ [41]. In [11], we investigate whether and how one can obtain optimal repairs if one restricts the output of the repair process to being ABoxes rather than qABoxes. In general, such optimal ABox repairs need not exist. The main contribution of the paper is an approach that can decide the existence of optimal ABox repairs in exponential time, and can compute all such repairs in case they exist. The papers [15, 16] consider error-tolerant reasoning based on optimal repairs and [1] compares optimal repairs with contractions from the area of belief change. Moreover, an approach to computing optimal repairs of \mathcal{EL} TBoxes is developed in [34].

In the second part of this paper we have presented new results on how to represent exponentially large repairs in a polynomial way and infinite repairs in a finite way. It would be interesting to see whether such approaches can also be extended to other settings. We conjecture that non-cycle-restricted TBoxes can still be tackled by using shell-unfoldings for the DLs considered in [13]. However, in [13] we also show that optimal repairs need not exist if the role

inclusions are not regular. It is unclear whether this problem can be overcome by an appropriate finite representation of infinite repairs. Another interesting topic for future research is to investigate whether finitely represented rational repairs can be used in practice.

Authors' Contributions. FB and FK contributed equally to the paper. PK ran the experiments and wrote the description of them. He also wrote a first version of the proof of Proposition 32.

Acknowledgements. This work has been supported by Deutsche Forschungsgemeinschaft (DFG) in projects 430150274 (Repairing Description Logic Ontologies) and 389792660 (TRR 248: Foundations of Perspicuous Software Systems).

References

1. Baader, F.: Optimal repairs in ontology engineering as pseudo-contractions in belief change. In: Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing (SAC '23), March 27–31, 2023, Tallinn, Estonia. pp. 983–990. Association for Computing Machinery (2023), <https://doi.org/10.1145/3555776.3577719>
2. Baader, F., Borgwardt, S., Morawska, B.: Extending unification in \mathcal{EL} towards general tboxes. In: Brewka, G., Eiter, T., McIlraith, S.A. (eds.) Principles of Knowledge Representation and Reasoning: Proceedings of the Thirteenth International Conference, KR 2012, Rome, Italy, June 10–14, 2012. AAAI Press (2012), <http://www.aaai.org/ocs/index.php/KR/KR12/paper/view/4491>
3. Baader, F., Borgwardt, S., Morawska, B.: SAT Encoding of Unification in \mathcal{ELH}_{R^+} w.r.t. Cycle-Restricted Ontologies. In: Gramlich, B., Miller, D., Sattler, U. (eds.) Automated Reasoning - 6th International Joint Conference, IJCAR 2012, Manchester, UK, June 26–29, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7364, pp. 30–44. Springer (2012), https://doi.org/10.1007/978-3-642-31365-3_5
4. Baader, F., Brandt, S., Lutz, C.: Pushing the \mathcal{EL} envelope. In: Kaelbling, L.P., Saffioti, A. (eds.) IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30 - August 5, 2005. pp. 364–369. Professional Book Center (2005), <http://ijcai.org/Proceedings/05/Papers/0372.pdf>
5. Baader, F., Brandt, S., Lutz, C.: Pushing the \mathcal{EL} envelope. In: Kaelbling, L.P., Saffioti, A. (eds.) Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005). pp. 364–369. Morgan Kaufmann, Los Altos, Edinburgh (UK) (2005)
6. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
7. Baader, F., Horrocks, I., Lutz, C., Sattler, U.: An Introduction to Description Logic. Cambridge University Press (2017), <https://doi.org/10.1017/9781139025355>
8. Baader, F., Koopmann, P., Kriegel, F.: Optimal repairs in the description logic \mathcal{EL} revisited. In: Proceedings of the 18th European Conference on Logics in Artificial Intelligence (JELIA 2023), September 20–22, 2023, Dresden, Germany. Lecture Notes in Computer Science, Springer (2023)

9. Baader, F., Koopmann, P., Kriegel, F., Nuradiansyah, A.: Computing optimal repairs of quantified ABoxes w.r.t. static \mathcal{EL} TBoxes. In: Platzer, A., Sutcliffe, G. (eds.) Automated Deduction - CADE 28 - 28th International Conference on Automated Deduction, Virtual Event, July 12-15, 2021, Proceedings. Lecture Notes in Computer Science, vol. 12699, pp. 309–326. Springer (2021), https://doi.org/10.1007/978-3-030-79876-5_18
10. Baader, F., Koopmann, P., Kriegel, F., Nuradiansyah, A.: Computing optimal repairs of quantified ABoxes w.r.t. static \mathcal{EL} TBoxes (extended version). LTCS-Report 21-01, Chair of Automata Theory, Institute of Theoretical Computer Science, Technische Universität Dresden, Dresden, Germany (2021), <https://doi.org/10.25368/2022.64>
11. Baader, F., Koopmann, P., Kriegel, F., Nuradiansyah, A.: Optimal ABox repair w.r.t. static \mathcal{EL} TBoxes: from quantified ABoxes back to ABoxes. In: 19th Extended Semantic Web Conference, ESWC 2022, Hersonissos, Greece, May 29 – June 2, 2022, Proceedings. Lecture Notes in Computer Science, vol. 13261, pp. 130–146. Springer (2022), https://doi.org/10.1007/978-3-031-06981-9_8
12. Baader, F., Koopmann, P., Kriegel, F., Nuradiansyah, A.: Optimal ABox repair w.r.t. static \mathcal{EL} TBoxes: from quantified ABoxes back to ABoxes (extended version). LTCS-Report 22-01, Chair of Automata Theory, Institute of Theoretical Computer Science, Technische Universität Dresden, Dresden, Germany (2022), <https://doi.org/10.25368/2022.65>
13. Baader, F., Kriegel, F.: Pushing optimal ABox repair from \mathcal{EL} towards more expressive Horn-DLs. In: Kern-Isberner, G., Lakemeyer, G., Meyer, T. (eds.) Proceedings of the 19th International Conference on Principles of Knowledge Representation and Reasoning, KR 2022, Haifa, Israel, July 31 – August 5, 2022. pp. 22–32 (2022), <https://doi.org/10.24963/kr.2022/3>
14. Baader, F., Kriegel, F., Nuradiansyah, A.: Privacy-preserving ontology publishing for \mathcal{EL} instance stores. In: Calimeri, F., Leone, N., Manna, M. (eds.) Logics in Artificial Intelligence - 16th European Conference, JELIA 2019, Rende, Italy, May 7-11, 2019, Proceedings. Lecture Notes in Computer Science, vol. 11468, pp. 323–338. Springer (2019), https://doi.org/10.1007/978-3-030-19570-0_21
15. Baader, F., Kriegel, F., Nuradiansyah, A.: Error-tolerant reasoning in the description logic \mathcal{EL} based on optimal repairs. In: Governatori, G., Turhan, A. (eds.) Rules and Reasoning - 6th International Joint Conference, RuleML+RR 2022, Virtual, September 26-28, 2022, Proceedings. Lecture Notes in Computer Science, vol. 13752, pp. 227–243. Springer (2022), https://doi.org/10.1007/978-3-031-21541-4_15
16. Baader, F., Kriegel, F., Nuradiansyah, A.: Treating role assertions as first-class citizens in repair and error-tolerant reasoning. In: Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing (SAC '23), March 27–31, 2023, Tallinn, Estonia. pp. 974–982. Association for Computing Machinery (2023), <https://doi.org/10.1145/3555776.3577630>
17. Baader, F., Kriegel, F., Nuradiansyah, A., Peñaloza, R.: Computing compliant anonymisations of quantified ABoxes w.r.t. \mathcal{EL} policies (extended version). LTCS-Report 20-08, Chair of Automata Theory, Institute of Theoretical Computer Science, Technische Universität Dresden, Dresden, Germany (2020), <https://doi.org/10.25368/2022.263>
18. Baader, F., Kriegel, F., Nuradiansyah, A., Peñaloza, R.: Making repairs in description logics more gentle. In: Thielscher, M., Toni, F., Wolter, F. (eds.) Principles of Knowledge Representation and Reasoning: Proceedings of the Sixteenth Interna-

- tional Conference, KR 2018, Tempe, Arizona, 30 October - 2 November 2018. pp. 319–328. AAAI Press (2018), <https://aaai.org/ocs/index.php/KR/KR18/paper/view/18056>
19. Baader, F., Kriegel, F., Nuradiansyah, A., Peñaloza, R.: Computing compliant anonymisations of quantified ABoxes w.r.t. \mathcal{EL} policies. In: Pan, J.Z., Tamma, V.A.M., d’Amato, C., Janowicz, K., Fu, B., Polleres, A., Seneviratne, O., Kagal, L. (eds.) *The Semantic Web - ISWC 2020 - 19th International Semantic Web Conference*, Athens, Greece, November 2-6, 2020, Proceedings, Part I. *Lecture Notes in Computer Science*, vol. 12506, pp. 3–20. Springer (2020), https://doi.org/10.1007/978-3-030-62419-4_1
 20. Baader, F., Suntisrivaraporn, B.: Debugging SNOMED CT using axiom pinpointing in the description logic \mathcal{EL}^+ . In: Cornet, R., Spackman, K.A. (eds.) *Proceedings of the Third International Conference on Knowledge Representation in Medicine*, Phoenix, Arizona, USA, May 31st - June 2nd, 2008. *CEUR Workshop Proceedings*, vol. 410. CEUR-WS.org (2008), <http://ceur-ws.org/Vol-410/Paper01.pdf>
 21. Bauslaugh, B.L.: Cores and compactness of infinite directed graphs. *J. Comb. Theory, Ser. B* **68**(2), 255–276 (1996), <https://doi.org/10.1006/jctb.1996.0068>
 22. Bauslaugh, B.L.: Compactness and finite equivalence of infinite digraphs. *Discret. Math.* **167-168**, 115–126 (1997), [https://doi.org/10.1016/S0012-365X\(96\)00220-8](https://doi.org/10.1016/S0012-365X(96)00220-8)
 23. Bauslaugh, B.L.: List-compactness of infinite directed graphs. *Graphs Comb.* **17**(1), 17–38 (2001), <https://doi.org/10.1007/s003730170052>
 24. Brachman, R.J., Fikes, R., Levesque, H.J.: Krypton: A functional approach to knowledge representation. *Computer* **16**(10), 67–73 (1983), <https://doi.org/10.1109/MC.1983.1654200>
 25. Carral, D., Dragoste, I., González, L., Jacobs, C.J.H., Krötzsch, M., Urbani, J.: Vlog: A rule engine for knowledge graphs. In: Ghidini, C., Hartig, O., Maleshkova, M., Svátek, V., Cruz, I.F., Hogan, A., Song, J., Lefrançois, M., Gandon, F. (eds.) *The Semantic Web - ISWC 2019 - 18th International Semantic Web Conference*, Auckland, New Zealand, October 26-30, 2019, Proceedings, Part II. *Lecture Notes in Computer Science*, vol. 11779, pp. 19–35. Springer (2019), https://doi.org/10.1007/978-3-030-30796-7_2
 26. Chandra, A.K., Merlin, P.M.: Optimal implementation of conjunctive queries in relational data bases. In: Hopcroft, J.E., Friedman, E.P., Harrison, M.A. (eds.) *Proc. of the 9th Annual ACM Symposium on Theory of Computing (STOC’77)*. pp. 77–90. ACM (1977), <https://doi.org/10.1145/800105.803397>
 27. Deutsch, A., Nash, A., Rummel, J.B.: The chase revisited. In: Lenzerini, M., Lembo, D. (eds.) *Proceedings of the Twenty-Seventh ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2008, June 9-11, 2008, Vancouver, BC, Canada*. pp. 149–158. ACM (2008), <https://doi.org/10.1145/1376916.1376938>
 28. Du, J., Qi, G., Fu, X.: A practical fine-grained approach to resolving incoherent OWL 2 DL terminologies. In: *Proc. of the 23rd ACM Int. Conf. on Information and Knowledge Management, (CIKM’14)*. pp. 919–928 (2014), <http://doi.acm.org/10.1145/2661829.2662046>
 29. Greiner, R., Smith, B.A., Wilkerson, R.W.: A correction to the algorithm in Reiter’s theory of diagnosis. *Artif. Intell.* **41**(1), 79–88 (1989), [https://doi.org/10.1016/0004-3702\(89\)90079-9](https://doi.org/10.1016/0004-3702(89)90079-9)
 30. Henzinger, M.R., Henzinger, T.A., Kopke, P.W.: Computing simulations on finite and infinite graphs. In: *36th Annual Symposium on Foundations of Computer Sci-*

- ence, Milwaukee, Wisconsin, USA, 23-25 October 1995. pp. 453–462. IEEE Computer Society (1995), <https://doi.org/10.1109/SFCS.1995.492576>
31. Horridge, M., Parsia, B., Sattler, U.: Laconic and precise justifications in OWL. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T.W., Thirunarayan, K. (eds.) The Semantic Web - ISWC 2008, 7th International Semantic Web Conference, ISWC 2008, Karlsruhe, Germany, October 26-30, 2008. Proceedings. Lecture Notes in Computer Science, vol. 5318, pp. 323–338. Springer (2008), https://doi.org/10.1007/978-3-540-88564-1_21
 32. Kazakov, Y., Krötzsch, M., Simancik, F.: The incredible ELK - from polynomial procedures to efficient reasoning with \mathcal{EL} ontologies. Journal of Automated Reasoning **53**(1), 1–61 (2014), <https://doi.org/10.1007/s10817-013-9296-3>
 33. Kriegel, F.: Navigating the \mathcal{EL} Subsumption Hierarchy. In: Homola, M., Ryzhikov, V., Schmidt, R.A. (eds.) Proceedings of the 34th International Workshop on Description Logics (DL 2021), Hybrid Event, Bratislava, Slovakia, September 19–22, 2021. CEUR Workshop Proceedings, vol. 2954. CEUR-WS.org (2021), <http://ceur-ws.org/Vol-2954/paper-21.pdf>
 34. Kriegel, F.: Optimal fixed-premise repairs of \mathcal{EL} TBoxes. In: Bergmann, R., Malburg, L., Rodermund, S.C., Timm, I.J. (eds.) Proceedings of the 45th German Conference on Artificial Intelligence (KI 2022), Virtual in Trier, Germany, September 19–23, 2022. Lecture Notes in Computer Science, vol. 13404, pp. 115–130. Springer (2022), https://doi.org/10.1007/978-3-031-15791-2_11
 35. Krötzsch, M., Marx, M., Rudolph, S.: The power of the terminating chase (invited talk). In: Barceló, P., Calautti, M. (eds.) 22nd International Conference on Database Theory, ICDT 2019, March 26-28, 2019, Lisbon, Portugal. LIPIcs, vol. 127, pp. 3:1–3:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2019), <https://doi.org/10.4230/LIPIcs.ICDT.2019.3>
 36. Lam, J.S.C., Sleeman, D.H., Pan, J.Z., Vasconcelos, W.W.: A fine-grained approach to resolving unsatisfiable ontologies. J. Data Semant. **10**, 62–95 (2008), https://doi.org/10.1007/978-3-540-77688-8_3
 37. Levesque, H.J.: Foundations of a functional approach to knowledge representation. Artif. Intell. **23**(2), 155–212 (1984), [https://doi.org/10.1016/0004-3702\(84\)90009-2](https://doi.org/10.1016/0004-3702(84)90009-2)
 38. Lutz, C., Seylan, I., Wolter, F.: An automata-theoretic approach to uniform interpolation and approximation in the description logic \mathcal{EL} . In: Brewka, G., Eiter, T., McIlraith, S.A. (eds.) Principles of Knowledge Representation and Reasoning: Proceedings of the Thirteenth International Conference, KR 2012, Rome, Italy, June 10-14, 2012. AAAI Press (2012), <http://www.aaai.org/ocs/index.php/KR/KR12/paper/view/4511>
 39. Lutz, C., Wolter, F.: Deciding inseparability and conservative extensions in the description logic \mathcal{EL} . J. Symb. Comput. **45**(2), 194–228 (2010), <https://doi.org/10.1016/j.jsc.2008.10.007>
 40. Motik, B., Cuenca Grau, B., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C.: OWL 2 web ontology language profiles (second edition). W3C recommendation (2012), <http://www.w3.org/TR/2012/REC-owl2-profiles-20121211/>
 41. Ortiz, M., Rudolph, S., Šimkus, M.: Worst-case optimal reasoning for the Horn-DL fragments of OWL 1 and 2. In: Lin, F., Sattler, U., Truszczyński, M. (eds.) Principles of Knowledge Representation and Reasoning: Proceedings of the Twelfth International Conference, KR 2010 (2010), <http://aaai.org/ocs/index.php/KR/KR2010/paper/view/1296>

42. Parsia, B., Matentzoglou, N., Gonçalves, R.S., Glimm, B., Steigmiller, A.: The OWL reasoner evaluation (ORE) 2015 competition report. *J. Autom. Reason.* **59**(4), 455–482 (2017), <https://doi.org/10.1007/s10817-017-9406-8>, test ontology corpus: <https://doi.org/10.5281/zenodo.18578>
43. Parsia, B., Rudolph, S., Hitzler, P., Krötzsch, M., Patel-Schneider, P.: OWL 2 web ontology language primer (second edition). W3C recommendation (2012), <http://www.w3.org/TR/2012/REC-owl2-primer-20121211/>
44. Parsia, B., Sirin, E., Kalyanpur, A.: Debugging OWL ontologies. In: Ellis, A., Hagino, T. (eds.) Proceedings of the 14th international conference on World Wide Web, WWW 2005, Chiba, Japan, May 10-14, 2005. pp. 633–640. ACM (2005), <https://doi.org/10.1145/1060745.1060837>
45. Reiter, R.: A theory of diagnosis from first principles. *Artif. Intell.* **32**(1), 57–95 (1987), [https://doi.org/10.1016/0004-3702\(87\)90062-2](https://doi.org/10.1016/0004-3702(87)90062-2), see the erratum [29].
46. Schlobach, S., Huang, Z., Cornet, R., van Harmelen, F.: Debugging incoherent terminologies. *J. Autom. Reason.* **39**(3), 317–349 (2007), <https://doi.org/10.1007/s10817-007-9076-z>
47. Troquard, N., Confalonieri, R., Galliani, P., Peñaloza, R., Porello, D., Kutz, O.: Repairing ontologies via axiom weakening. In: McIlraith, S.A., Weinberger, K.Q. (eds.) Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018. pp. 1981–1988. AAAI Press (2018), <https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17189>
48. Tychonoff, A.N.: Über einen Funktionenraum. *Mathematische Annalen* **111**(1), 762–766 (1935), <https://doi.org/10.1007/BF01472255>
49. Zermelo, E.: Beweis, daß jede Menge wohlgeordnet werden kann. *Mathematische Annalen* **59**(4), 514–516 (1904), <https://doi.org/10.1007/BF01445300>