Optimal Alignment of Temporal Knowledge Bases

Oliver Fernández Gil [a,c;*], **Fabio Patrizi** [b;**], **Giuseppe Perelli** [b;***] **and Anni-Yasmin Turhan** [a,c;****]

[a]Theoretical Computer Science, Technische Universität Dresden, Germany
[b]Sapienza University of Rome, Italy
[c]Center for Scalable Data Analytics and Artificial Intelligence (ScaDS.AI) Dresden/Leipzig, Germany

Abstract. Answering temporal CQs over temporalized Description Logic knowledge bases (TKB) is a main technique to realize ontology-based situation recognition. In case the collected data in such a knowledge base is inaccurate, important query answers can be missed. In this paper we introduce the TKB Alignment problem, which computes a variant of the TKB that minimally changes the TKB, but entails the given temporal CQ and is in that sense (cost-)optimal. We investigate this problem for $\mathcal{ALC}$ TKBs and conjunctive queries with LTL operators and devise a solution technique to compute (cost-optimal) alignments of TKBs that extends techniques for the alignment problem for propositional LTL over finite traces.

1 Introduction

Observing complex systems over time and drawing conclusions about their behavior is a core task for many AI systems. In particular, adaptive systems have to recognize situations in which an adaptation is useful. A well-investigated approach to do this is ontology-based situation recognition [3, 1, 24]. This approach is usually realized by modeling the observed system by a temporal knowledge base (TKB), where the data from the observed system is collected over time and stored in a sequence of ABoxes and a TBox that models important notions from the application domain. The situation to be recognized by the system is then modeled by a temporal query that will be answered over the sequence of ABoxes and the TBox. The situation recognition is then to detect predefined situations that are formalized as temporal (conjunctive) queries over the observed and enriched ABox sequence. As in classical ontology-mediated query answering [7], the TBox enriches the data in the ABox sequence as it restricts its interpretation and allows for more conclusions. The semantics of TKBs is given by an infinite sequence of first-order interpretations. TKBs can be queried by temporal conjunctive queries (TCQs), which combine LTL with conjunctive queries. Methods for answering temporal queries over TKBs and testing entailment of Boolean TCQs have been intensively investigated ([4, 8, 1]).

Now, in many applications, the data is collected from several sources and need not always be accurate. Consider the medical domain, where deviations of classical symptoms are frequent for certain patient groups or where examination methods such as blood test results can be inaccurate or discretized unsuitably. Thus the query need not return the expected answer, although the patient or, in the general case, the observed system is in a critical state that requires adaptation. The problem is to find a version of the TKB that admits to detect "near misses".

There are mainly two approaches developed to address the problem of errors or inaccuracies in DL knowledge bases. In case of inconsistent TKBs, ontology repairs restore consistent versions by deleting statements from the ABoxes [9, 6]. In case that information is missing in the ABoxes for the query to return answers, ABox abduction, i.e., adding new statements to the ABoxes has been investigated—mostly in the atemporal setting [10, 14, 18].

In this paper, we investigate the new task of TKB Alignment, i.e., to modify the sequence of ABoxes by deletions or additions of statements so as to yield answers for the TCQ. Surprisingly, this problem has not been addressed in the literature yet. The goal of this paper is to develop an approach to solve instances of this new problem.

The well-known problem of Trace Alignment realizes a very similar task to TKB alignment: for a finite trace of observations and a property specification expressed in Linear Temporal Logic (LTL), a minimal modification of the trace is produced that satisfies the specification. This task has been extensively studied by the Business Process (BP) and AI communities, leading to effective solutions and implemented tools; see, e.g., [11, 13, 12]. In all these settings, the observations recorded in a trace are propositional, i.e., each time point of the trace represents one of finitely many possible observables, modeled as propositions.

In this paper, we address the problem of TKB Alignment as a Trace Alignment problem in a much richer setting, where observables are described by DL concepts and roles, and properties are specified by a temporalized query using DL atoms. Furthermore, the open world semantics of DLs is adopted, since entailment is considered instead of satisfaction as in classical propositional trace alignment.

We investigate the following setting for TKB alignment: a TCQ using (future) LTL operators and a TKB written in the DL $\mathcal{ALC}$, together with a cost measure for edit operations on the ABox sequence. Solving TKB alignment is then to compute an ABox sequence which, together with the TBox, entails the Boolean TCQ, while guaranteeing cost-optimality of the modification. Intuitively, the cost-optimal version of the TKB states which minimal changes of the TKB would result in answers to the TCQ.

The technique we develop builds on an approach for deciding temporal query entailment over TKBs by [4] and one for solving LTL Trace Alignment for finite traces by [11], and extends them non-trivially. Our technique extends the former approach from verification to synthesis and the latter from the propositional to the DL set-

* Corresponding Author. Email: oliver.fernandez@tu-dresden.de
** Email: patrizi@diag.uniroma1.it
*** Email: perelli@di.uniroma1.it
**** Email: anni-yasmin.turhan@tu-dresden.de

ting, from propositional traces to TKBs, and from finite to infinite traces. This ultimately results in an effective solution approach which can assess the deviation of irregular observations wrt standard ones and define corrective actions to recover a standard observation.

The detailed proofs for all results can be found in [15].

2 Preliminaries

In this section we recap basic notions on description logics, LTL, and trace alignment.

2.1 Description Logic Knowledge Bases

Description Logics (DLs) are a family of formal languages for representing knowledge and reasoning about it. In this work, we focus on the DL $\mathcal{ALC}$ ([23]).

We fix three countably infinite sets of names: $\mathsf{N_C}$ for concepts, $\mathsf{N_R}$ for roles and $\mathsf{N_I}$ for individuals. *Concepts* in $\mathcal{ALC}$ are defined inductively as follows:

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

where $A \in \mathsf{N_C}$, $r \in \mathsf{N_R}$, $\top$ is the *top-concept* and $\bot$ the *bottom-concept*. We use the following standard abbreviations: $C \sqcap D$ for $\neg(\neg C \sqcup \neg D)$, $\forall r.C$ for $\neg(\exists r.\neg C)$, and $\bot$ for $\neg\top$.

DL concepts are interpreted over (first-order, FO) interpretations. An *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a domain $\Delta^{\mathcal{I}}$ and a function $\cdot^{\mathcal{I}}$ mapping each concept name A to a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, each role name r to a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, and each individual name a to an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. Based on this, the semantics of (complex) concepts is defined as follows: $(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$, $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$, $(\exists r.C)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \exists e.((d,e) \in r^{\mathcal{I}} \wedge e \in C^{\mathcal{I}})\}$, and $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$.

General concept inclusions (GCIs) are statements of the form $C \sqsubseteq D$, expressing inclusion relationships between concepts. A *TBox* (denoted $\mathcal{T}$) is a finite set of GCIs. A *model* of a TBox $\mathcal{T}$ is an interpretation $\mathcal{I}$ that satisfies all GCIs in $\mathcal{T}$, i.e., for all $C \sqsubseteq D \in \mathcal{T}$, it holds that $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. A TBox is *satisfiable* if it has a model.

Statements $A(a)$ and $r(a,b)$ are called, respectively, *concept assertion* and *role assertion*, where $a, b \in \mathsf{N_I}$, $A \in \mathsf{N_C}$ and $r \in \mathsf{N_R}$. An interpretation $\mathcal{I}$ satisfies $A(a)$ if $a^{\mathcal{I}} \in A^{\mathcal{I}}$, and satisfies $r(a,b)$ if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$. An *ABox* $\mathcal{A}$ is a finite set of (concept or role) assertions. An interpretation $\mathcal{I}$ is a *model* of an ABox $\mathcal{A}$, if $\mathcal{I}$ satisfies all assertions in $\mathcal{A}$.

A *DL knowledge base* (KB) is a pair $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, with $\mathcal{T}$ a TBox and $\mathcal{A}$ an ABox. An interpretation $\mathcal{I}$ is a *model* of $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, written $\mathcal{I} \models \mathcal{K}$, if $\mathcal{I}$ is a model of $\mathcal{T}$ and $\mathcal{A}$. A KB is *consistent* if it has a model.

2.2 Conjunctive Queries

Prominent reasoning problems investigated in the last decade concern *conjunctive queries*. We briefly recap related definitions and results.

Definition 1 (Conjunctive query) *Let $\mathsf{N_V}$ be a set of variables. A* conjunctive query (CQ) *is an expression of the form $\phi = \exists \bar{y}.\psi$, where $\bar{y}$ is a tuple of variables from $\mathsf{N_V}$ and ψ is a finite conjunction of* atoms *of the form: $A(z)$, for $A \in \mathsf{N_C}$ and $z \in \mathsf{N_V} \cup \mathsf{N_I}$, or $r(z,z')$, for $r \in \mathsf{N_R}$ and $z, z' \in \mathsf{N_V} \cup \mathsf{N_I}$.*

By $\textsc{cq}$ we denote the set of all CQs (over $\mathsf{N_C}$, $\mathsf{N_R}$, $\mathsf{N_I}$, $\mathsf{N_V}$). We write $A(z) \in \phi$ to state that atom $A(z)$ occurs in ϕ, and likewise for $r(z,z')$. In this work, we combine CQs using Boolean connectives.

Definition 2 (Boolean combination of CQs) *A formula ϕ is a* Boolean combination of CQs *iff:*

$$\phi := \phi' \mid \neg\phi \mid \phi \vee \phi, \textit{ where } \phi' \in \textsc{cq}.$$

As standard, $\phi_1 \wedge \phi_2$ abbreviates $\neg(\neg\phi_1 \vee \neg\phi_2)$. Given a Boolean combination of CQs ϕ, we denote by $\mathsf{Var}(\phi)$, $\mathsf{FVar}(\phi)$, and $\mathsf{Ind}(\phi)$ the set of variables, free variables and individual names occurring in ϕ, respectively. A query with no free variables is called *Boolean*, whereas a query with $\mathsf{Ind}(\phi) = \emptyset$ is called *pure*. $\textsc{bcq}$ denotes the set of *Boolean CQs* (BCQs), $\mathcal{B}(\textsc{cq})$ the set of Boolean combinations of CQs, and $\mathcal{B}(\textsc{bcq})$ the set of Boolean combinations of BCQs.

The semantics of BCQs is defined in terms of a satisfaction relation between interpretations and BCQs.

Definition 3 (Semantics of BCQs) *An interpretation $\mathcal{I}$ is* a model *of (or* satisfies*) a BCQ ϕ, written $\mathcal{I} \models \phi$, iff there exists a mapping $h : \mathsf{Var}(\phi) \cup \mathsf{Ind}(\phi) \to \Delta^{\mathcal{I}}$, called a* match*, s.t.:*

- *$h(a) = a^{\mathcal{I}}$ for all $a \in \mathsf{Ind}(\phi)$;*
- *$h(z) \in A^{\mathcal{I}}$ for all $A(z) \in \phi$; and*
- *$(h(z), h(z')) \in r^{\mathcal{I}}$ for all $r(z,z') \in \phi$.*

These notions straightforwardly extend to $\mathcal{B}(\textsc{bcq})$.

Definition 4 (Semantics of Boolean combinations of BCQs) *An interpretation $\mathcal{I}$ is* a model *of (or* satisfies*) a query $\phi \in \mathcal{B}(\textsc{bcq})$, written $\mathcal{I} \models \phi$, iff:*

- *$\phi \in \textsc{bcq}$ and $\mathcal{I} \models \phi$; or*
- *$\phi = \neg\phi_1$ and $\mathcal{I} \not\models \phi_1$; or*
- *$\phi = \phi_1 \vee \phi_2$ and $\mathcal{I} \models \phi_1$ or $\mathcal{I} \models \phi_2$.*

A query $\phi \in \mathcal{B}(\textsc{bcq})$ is *satisfiable* wrt a KB $\mathcal{K}$, if $\mathcal{I} \models \phi$ for some model $\mathcal{I}$ of $\mathcal{K}$. A knowledge base $\mathcal{K}$ *entails* a query ϕ (written $\mathcal{K} \models \phi$), if $\mathcal{I} \models \phi$ for all models $\mathcal{I}$ of $\mathcal{K}$.

In case of non-Boolean queries, one is interested in computing the certain answers. More precisely, given a KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ and $\phi \in \mathcal{B}(\textsc{cq})$ with free variables $\bar{x} = (x_1, \ldots, x_k)$, a tuple $\bar{a} = (a_1, \ldots, a_k)$ of individuals in $\mathsf{N_I}$ is a *certain answer of ϕ wrt $\mathcal{K}$* if $\mathcal{K} \models \phi[\bar{a}]$, where $\phi[\bar{a}]$ is the Boolean query obtained from ϕ by replacing each occurrence of x_i by a_i ($1 \leq i \leq k$). We denote by $cert_{\mathcal{K}}(\phi)$ the set of certain answers of ϕ wrt $\mathcal{K}$. If ϕ is a Boolean query and $\mathcal{K} \models \phi$, then $cert_{\mathcal{K}}(\phi) = \{()\}$.

The entailment problem for BCQs wrt $\mathcal{ALC}$ knowledge bases is ExpTime-complete ([20, 21]). It was shown in [4] for $\mathcal{B}(\textsc{bcq})$ that satisfiability of a conjunction of CQ-literals (i.e. either a Boolean CQ or a negated Boolean CQ) wrt $\mathcal{ALC}$ knowledge bases is an ExpTime-complete problem. An easy consequence of this (and of $\mathcal{B}(\textsc{bcq})$ being *closed under negation*) is that satisfiability and entailment of arbitrary Boolean combinations of BCQs w.r.t. $\mathcal{ALC}$ knowledge bases are also ExpTime-complete problems.

2.3 Propositional Linear Temporal Logic

The kind of properties we focus on in this paper concerns the evolution of a knowledge base over time. To express relevant properties, we need a temporal logic. We review the basics on propositional Linear Temporal Logic ($\textsc{ltl}$), which will be later lifted to CQs and used on to address TKB Alignment.

The language of Linear Temporal Logic (LTL) formulas φ is defined over a finite set of propositions PROP, as follows:

$$\varphi := p \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathbf{X}\,\varphi \mid \varphi\,\mathbf{U}\,\varphi, \text{ with } p \in \text{PROP}.$$

The set $\mathsf{props}(\varphi)$ denotes the finite set of propositions occurring in φ. LTL formulas are interpreted over infinite words, also called *(propositional) traces*, $w = w_0 w_1 \cdots \in (2^{\text{PROP}})^\omega$.

Definition 5 (LTL semantics) *Given a formula $\varphi \in$ LTL, a trace $w = w_0 w_1 \cdots$, and an index i, we inductively define when w, i satisfy φ, denoted $w, i \models \varphi$, as follows:*

- *$w, i \models p$, if $p \in w_i$;*
- *$w, i \models \neg\varphi$, if $w, i \not\models \varphi$;*
- *$w, i \models \varphi_1 \vee \varphi_2$, if $w, i \models \varphi_1$ or $w, i \models \varphi_2$;*
- *$w, i \models \mathbf{X}\,\varphi$, if $w, i+1 \models \varphi$;*
- *$w, i \models \varphi_1\,\mathbf{U}\,\varphi_2$ if there exists $j \geq i$ s.t. $w, j \models \varphi_2$ and $w, k \models \varphi_1$, for $k = i, \ldots, j-1$.*

We say that w satisfies *$\varphi \in$ LTL, written $w \models \varphi$, iff $w, 0 \models \varphi$.*

We denote the set of traces satisfying φ as $\mathcal{L}(\varphi) = \{w \in (2^{\text{PROP}})^\omega \mid w \models \varphi\}$. It is well-known that for every $\varphi \in$ LTL there exists a deterministic parity automaton P_φ accepting exactly $\mathcal{L}(\varphi)$.

A *deterministic parity automaton* (DPA) is a tuple $P = (Al, Q, \delta, q_0, \mathsf{col})$, where: Al is the finite input alphabet, Q is the finite set of states, $\delta : Q \times Al \to Q$ is the transition function, $q_0 \in Q$ is the initial state, and $\mathsf{col} : Q \to Col$ is a *coloring function*, mapping the states of P into a finite set of *colors* $Col \subset \mathbb{N}_0$. DPAs are similar to deterministic finite-state automata (DFA), but accept *infinite* traces and thus have a different accepting condition.

For a DPA P, a *finite run from state* $q \in Q$ is a sequence $\rho = q \overset{w_0}{\to} q_1 \overset{w_1}{\to} \cdots \overset{w_{n-1}}{\to} q_n$ s.t. $\delta(q, w_0) = q_1$ and $\delta(q_i, w_i) = q_{i+1}$, for $0 < i < n$. We define *infinite* runs analogously, for $n = \infty$. Unless stated otherwise, runs are always infinite and start in the initial state q_0 of P.

Given a run ρ of P, let $\mathsf{inf}_Q(\rho, P)$ be the set of states occurring infinitely many times in ρ. Obviously, $\mathsf{inf}_Q(\rho, P) \neq \emptyset$ iff ρ is infinite. Let $\mathsf{inf}(\rho, P) = \{\mathsf{col}(q) \in Col \mid q \in \mathsf{inf}_Q(\rho, P)\}$ be the set of colors "visited" infinitely many times by ρ. A run ρ from a state $q \in Q$ is *accepting* iff $\min\{\mathsf{inf}(\rho, P)\}$ is even. When this is the case, $q \in Q$ is an *accepting* state. By $\mathsf{Acc}(P)$, we denote the set of all accepting states of P and call $\mathsf{Acc}(P)$ the *accepting* set of P.

Lemma 1 ([17]) *The accepting set* $\mathsf{Acc}(P)$ *of a DPA $P = (Al, Q, \delta, q_0, \mathsf{col})$ can be computed in time $(|Q| + |\delta|)\log|Col|$, where $|\delta| = |\{(q, q') \in Q \times Q$ s.t. $q' = \delta(a, q)$, for some $a\}|$.*

For a DPA $P = (Al, Q, \delta, q_0, \mathsf{col})$ and a trace $w = w_0 w_1 \cdots$, the (unique) run *induced* by w is the run $\rho = q_0 \overset{w_0}{\to} q_1 \overset{w_1}{\to} \cdots$. A trace w is *accepted* by P iff the run ρ induced by w is accepting. By $\mathcal{L}(P)$ we denote the *language* of P, i.e., the set of all traces accepted by P.

Theorem 1 ([25, 22]) *For every $\varphi \in$ LTL there exists a DPA $P_\varphi = (2^{\text{PROP}}, Q, \delta, q_0, \mathsf{col})$ s.t. $\mathcal{L}(P_\varphi) = \mathcal{L}(\varphi)$. P_φ can be computed in doubly exponential time and has doubly exponential size wrt φ.*

2.4 *Entailment of Temporal Conjunctive Queries*

TKB Alignment is closely related to checking *temporal query entailment* (TQE), studied in [4]. We briefly recall the main definitions and results from that work, possibly adapted to our setting.

Definition 6 (Temporal Knowledge Base (TKB)) *A* temporal knowledge base *(TKB) is a pair $\Gamma = (\mathcal{T}, \Lambda)$, where $\mathcal{T}$ is a TBox and $\Lambda = \mathcal{A}_0 \cdots \mathcal{A}_\ell$ is a finite sequence of ABoxes.*

A *FO trace* is an infinite sequence $\mathfrak{I} = \mathcal{I}_0 \mathcal{I}_1 \cdots$ of interpretations $\mathcal{I}_i = (\Delta, \cdot^{\mathcal{I}_i})$ over a fixed domain Δ.

Definition 7 (Model of a TKB) *Given a TKB $\Gamma = (\mathcal{T}, \mathcal{A}_0 \cdots \mathcal{A}_\ell)$, a FO trace $\mathfrak{I} = \mathcal{I}_0 \mathcal{I}_1 \cdots$ is a* model *of Γ, iff: $\mathcal{I}_i \models \mathcal{A}_i$, for $0 \leq i \leq \ell$; and $\mathcal{I}_i \models \mathcal{T}$ for all $i \geq 0$. If these conditions hold, $\mathfrak{I}$* satisfies *Γ, written $\mathfrak{I} \models \Gamma$.*

Next, we introduce the language TCQ of *(simple) temporal conjunctive queries* (TCQ), which essentially lifts propositions in LTL to BCQs. TCQ formulas are obtained as in LTL, by replacing PROP with a finite set PBCQ $\subset$ BCQ:

$$\varphi := \phi \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathbf{X}\,\varphi \mid \varphi\,\mathbf{U}\,\varphi, \text{ with } \phi \in \text{PBCQ}.$$

TCQ formulas are evaluated over FO traces.

Definition 8 (TCQ Semantics) *Given a formula $\varphi \in$ TCQ, an infinite FO trace $\mathfrak{I} = \mathcal{I}_0 \mathcal{I}_1 \cdots$, and an index i, we inductively define when $\mathfrak{I}$* satisfies *φ from i, denoted $\mathfrak{I}, i \models \varphi$, as follows:*

- *$\mathfrak{I}, i \models \phi$, if $\mathcal{I}_i \models \phi$, for $\phi \in$ PBCQ;*
- *$\mathfrak{I}, i \models \neg\varphi$ if $\mathfrak{I}, i \not\models \varphi$;*
- *$\mathfrak{I}, i \models \varphi_1 \vee \varphi_2$, if $\mathfrak{I}, i \models \varphi_1$ or $\mathfrak{I}, i \models \varphi_2$;*
- *$\mathfrak{I}, i \models \mathbf{X}\,\varphi$, if $\mathfrak{I}, i+1 \models \varphi$;*
- *$\mathfrak{I}, i \models \varphi_1\,\mathbf{U}\,\varphi_2$ if there exists $j \geq i$ s.t. $\mathfrak{I}, j \models \varphi_2$ and $\mathfrak{I}, k \models \varphi_1$, for $k = i, \ldots, j-1$.*

We say that $\mathfrak{I}$ satisfies *φ, written $\mathfrak{I} \models \varphi$, if $\mathfrak{I}, 0 \models \varphi$.*

The semantics of temporal operators for TCQs is analogous to that of LTL (see Def. 5); however, for TCQs the base case accounts for the satisfaction of BCQs by the FO interpretations occurring in the trace.

We observe that the variant of TCQs we use here differs from that introduced in [4], in that we disallow *past* operators ([16]). However, such restriction comes without loss of generality. This is because the two semantic variations have essentially the same expressive power, as future operators have the ability to mimic the past ones ([26, 19]).

The notion of *temporal conjunctive query entailment* used here is the same as that in [4], once past operators are disallowed in TCQs.

Definition 9 (TCQ Entailment) *Given a TKB Γ and a TCQ $\varphi \in$ TCQ. The TKB Γ* entails *φ, written $\Gamma \models \varphi$, iff $\mathfrak{I} \models \varphi$, for every model $\mathfrak{I}$ of Γ.*

Checking TCQ entailment (TQE) is the problem of deciding whether $\Gamma \models \varphi$.

3 The TKB Alignment Problem

We generalize the verification problem of TQE to a *synthesis* version, consisting in *minimally* modifying the sequence Λ of a TKB $\Gamma = (\mathcal{T}, \Lambda)$, to obtain a TKB $\Gamma' = (\mathcal{T}, \Lambda')$, s.t. $\Gamma' \models \varphi$. Observe that if $\Gamma \models \varphi$, the problem amounts to checking TQE. To define the problem, we formalize next the notions of ABox- and TKB-modification, and minimality.

To modify the ABoxes occurring in a TKB, we consider two kinds of ABox operations, namely *insertion* and *removal* of a (concept or role) assertion α, respectively denoted as $\mathrm{ins}(\alpha)$ and $\mathrm{rem}(\alpha)$. The result of applying such operations to an ABox $\mathcal{A}$ is given by the function mod: *(i)* $\mathsf{mod}(\text{ins}(\alpha), \mathcal{A}) = \mathcal{A} \cup \{\alpha\}$; *(ii)* $\mathsf{mod}(\text{rem}(\alpha), \mathcal{A}) = \mathcal{A} \setminus \{\alpha\}$. An *ABox-modification* is a (possibly empty) sequence $\mu = m_0 \cdots m_n$ of ABox operations m_i. By ε we denote the empty ABox-modification. The semantics of applying an ABox-modification μ to an ABox $\mathcal{A}$ is obtained by inductively extending mod to sequences of operations: *(i)* $\mathsf{mod}(\varepsilon, \mathcal{A}) = \mathcal{A}$; *(ii)* $\mathsf{mod}(m \cdot \mu, \mathcal{A}) = \mathsf{mod}(\mu, \mathsf{mod}(m, \mathcal{A}))$, where $\cdot$ is the concatenation operator.

We assume every operation m has a strictly positive cost $\mathsf{c}(m) \in \mathbb{R}^+$. The *cost* of an ABox-modification $\mu = m_1 \cdots m_n$ is defined as $\mathsf{c}(\mu) := \sum_{i=1}^{n} \mathsf{c}(m_i)$, with $\mathsf{c}(\varepsilon) = 0$.

In addition to modifying its ABoxes, a TKB can be modified by adding or removing ABoxes. Let $\mathsf{Atm} = \{(\text{fix}, \mu), (\text{add}, \mu), \text{del} \mid \mu \text{ is an ABox-modification}\}$ be the set of *atomic TKB-modifications*. Intuitively, (fix, μ) stands for the modification of an ABox through the application of μ, (add, μ) for the addition of the empty ABox followed by the application of μ, and del for ABox deletion.

A *TKB-modification* is a finite sequence η of atomic TKB-modifications, with ε denoting the empty TKB-modification. Notice that, by a slight abuse of notation, we use ε to denote both the empty ABox-modification and the empty TKB-modification; the intended meaning is clear from the context.

The result of applying a TKB-modification η to a sequence $\Lambda = \mathcal{A}_0 \cdots \mathcal{A}_\ell$ is the sequence $\mathsf{mod}(\eta, \Lambda)$, inductively defined as follows, where Λ_ϵ denotes the empty sequence of ABoxes, $\mathcal{A}_\emptyset$ the empty ABox, and $\mathcal{A}|\Lambda = \mathcal{A}\mathcal{A}_0 \cdots \mathcal{A}_\ell$:

- $\mathsf{mod}(\varepsilon, \Lambda) = \Lambda$;
- $\mathsf{mod}((\text{fix}, \mu), \Lambda_\epsilon) = \Lambda_\epsilon$;
- $\mathsf{mod}((\text{fix}, \mu) \cdot \eta, \mathcal{A}|\Lambda) = \mathsf{mod}(\mu, \mathcal{A})|\mathsf{mod}(\eta, \Lambda)$;
- $\mathsf{mod}((\text{add}, \mu) \cdot \eta, \Lambda) = \mathsf{mod}(\mu, \mathcal{A}_\emptyset)|\mathsf{mod}(\eta, \Lambda)$;
- $\mathsf{mod}(\text{del} \cdot \eta, \Lambda_\epsilon)) = \mathsf{mod}(\eta, \Lambda_\epsilon)$;
- $\mathsf{mod}(\text{del} \cdot \eta, \mathcal{A}|\Lambda) = \mathsf{mod}(\eta, \Lambda)$.

For a given TKB $\Gamma = (\mathcal{T}, \Lambda)$ and a TKB-modification η, we define $\mathsf{mod}(\eta, \Gamma) = (\mathcal{T}, \mathsf{mod}(\eta, \Lambda))$. The cost function naturally extends to TKB-modifications and ABox sequences:

- $\mathsf{c}(\varepsilon, \Lambda) = 0$;
- $\mathsf{c}((\text{fix}, \mu) \cdot \eta, \Lambda_\epsilon) = \mathsf{c}(\eta, \Lambda_\epsilon)$;
- $\mathsf{c}((\text{fix}, \mu) \cdot \eta, \mathcal{A}|\Lambda) = \mathsf{c}(\mu) + \mathsf{c}(\eta, \Lambda)$;
- $\mathsf{c}((\text{add}, \mu) \cdot \eta, \Lambda) = 1 + \mathsf{c}(\mu) + \mathsf{c}(\eta, \Lambda)$;
- $\mathsf{c}(\text{del} \cdot \eta, \Lambda_\epsilon) = \mathsf{c}(\eta, \Lambda_\epsilon)$;
- $\mathsf{c}(\text{del} \cdot \eta, \mathcal{A}|\Lambda) = \left(\sum_{\alpha \in \mathcal{A}} \mathsf{c}(\text{rem}(\alpha))\right) + 1 + \mathsf{c}(\eta, \Lambda)$.

The cost of (add, μ) is that of adding the empty ABox (taken as 1) and applying μ to it; similarly, the cost of del is that of emptying the ABox, by removing all of its assertions, and removing the resulting empty ABox (i.e., 1). For a TKB $\Gamma = (\mathcal{T}, \Lambda)$ and a TKB-modification η, we let $\mathsf{c}(\eta, \Gamma) = \mathsf{c}(\eta, \Lambda)$.

We can now introduce the TKB-alignment problem.

Definition 10 (TKB Alignment) *Given a TKB Γ and a TCQ φ, the* TKB-alignment *problem consists in finding a minimal-cost TKB-modification η^* (if any) s.t. $\mathsf{mod}(\eta^*, \Gamma) \models \varphi$.*

Observe that, for every TKB-modification η and TKB Γ, it holds that $\mathsf{mod}(\eta, \Gamma) = \mathsf{mod}(\eta \cdot (\text{fix}, \epsilon), \Gamma)$ and $\mathsf{c}(\eta, \Gamma) = \mathsf{c}(\eta \cdot (\text{fix}, \epsilon), \Gamma)$. That is, appending a sequence of (fix, ϵ) to η does not affect the result or the cost of modifying Γ. Thus, we can always extend η to guarantee that the combined number of occurrences of deletions del and fixes (fix, μ) in η equals at least the number of ABoxes in Λ. For technical convenience, from now on, we assume this is the case for every η.

3.1 Solving TKB Alignment

Our solution approach consists in reducing TKB Alignment to Shortest Path. To this end, we construct a graph, called *Minimal-instantiation Graph*, with each edge labelled by an atomic TKB-modification and its corresponding cost, s.t. every shortest path from a suitably defined *initial* node to one node from a (suitably defined) *target* set, represents an optimal solution to the original TKB Alignment instance. The construction of such a graph is based on several intermediate structures. In the following, for each of such structures and related notions, we first report their formal definitions and then give intuitive explanations for these.

Consider a TKB $\Gamma = (\mathcal{T}, \Lambda)$ and a TCQ φ. We start with the construction of a DPA intended to accept the set of models of φ. To this end, we adopt an approach similar to that of [4], which uses the *propositional abstraction* of φ. If we view every BCQ $\phi \in$ PBCQ as a proposition $\hat{\phi}$, then φ can be viewed as an LTL formula. This is called the *propositional abstraction* of φ, denoted $\hat{\varphi}$. Obviously, $\mathsf{props}(\hat{\varphi})$ is the set of all propositions $\hat{\phi}$ occurring in $\hat{\varphi}$, each corresponding to exactly one BCQ $\phi \in$ PBCQ.

Since $\hat{\varphi} \in$ LTL, we can now use the Büchi automaton (BA) construction of [25], to obtain a BA that recognizes $\mathcal{L}(\hat{\varphi})$ and then use the BA-to-DPA construction of [22] to obtain the DPA $P_{\hat{\varphi}} = (2^{\mathsf{props}(\hat{\varphi})}, Q, q_0, \delta, \mathsf{col})$ of $\hat{\varphi}$. The importance of $P_{\hat{\varphi}}$ lies in the fact that, although it reads input words $\Psi = \Phi_0\Phi_1 \cdots \in (2^{\mathsf{props}(\hat{\varphi})})^\omega$ and not FO traces $\mathfrak{I} = \mathcal{I}_0\mathcal{I}_1 \cdots$, it fully characterizes $\mathcal{L}(\varphi)$, as discussed below.

For $\Phi \in 2^{\mathsf{props}(\hat{\varphi})}$, let $\chi(\Phi) = \bigwedge_{\hat{\phi} \in \Phi} \phi \wedge \bigwedge_{\hat{\phi} \in \mathsf{props}(\hat{\varphi}) \setminus \Phi} \neg\phi$. Since the conjuncts in $\chi(\Phi)$ are possibly negated BCQs from PBCQ, and not propositional abstractions, $\chi(\Phi) \in \mathcal{B}(\text{BCQ})$. If $\chi(\Phi)$ is consistent, i.e., admits at least one model, Φ is called a *type*. When $\mathcal{I} \models \chi(\Phi)$, we call Φ the *type* of $\mathcal{I}$. This notion naturally extends to traces by defining the *trace type* of a FO trace $\mathfrak{I} = \mathcal{I}_0\mathcal{I}_1 \cdots$ as the word $\Psi = \Phi_0\Phi_1 \cdots \in (2^{\mathsf{props}(\hat{\varphi})})^\omega$ s.t. Φ_i is the type of $\mathcal{I}_i$, for all $i \geq 0$. We have the following result.

Lemma 2 *Every FO interpretation has a unique type and every type admits an FO interpretation. Moreover, every FO trace has a unique trace type and every trace type admits an FO trace.*

The following result relates $\mathcal{L}(P_{\hat{\varphi}})$ and $\mathcal{L}(\varphi)$.

Theorem 2 *Consider a TCQ φ. For every FO trace $\mathfrak{I}$ of type Ψ, it holds that $\mathfrak{I} \in \mathcal{L}(\varphi)$ iff $\Psi \in \mathcal{L}(P_{\hat{\varphi}})$.*

Since $P_{\hat{\varphi}}$ is independent of the TKB Γ, it cannot be used to search for the desired minimal-cost modification. For this, we can use a deterministic finite-state automaton (DFA) D, called the *repair-template* DFA for Γ and φ. The definition of D requires an auxiliary DPA, called the $\mathcal{T}$*-reduct* of $P_{\hat{\varphi}}$, to define the final states of D.

Definition 11 ($\mathcal{T}$-reduct of $P_{\hat{\varphi}}$) *Given a TBox $\mathcal{T}$ and a TCQ φ, let $P_{\hat{\varphi}} = (2^{\mathsf{props}(\hat{\varphi})}, Q, \delta, q_0, \mathsf{col})$. The $\mathcal{T}$-reduct of $P_{\hat{\varphi}}$ is the DPA $P_{\hat{\varphi}}^{\mathcal{T}} = (2^{\mathsf{props}(\hat{\varphi})}, Q^{\mathcal{T}}, \delta^{\mathcal{T}}, q_0, \mathsf{col}^{\mathcal{T}})$ s.t.:*

- *$Q^{\mathcal{T}} = Q \cup \{q^*\}$, with $q^* \notin Q$;*
- *$\delta^{\mathcal{T}}(q, \Phi) = q'$, iff either:*
  - *$\chi(\Phi)$ is satisfiable wrt $\mathcal{T}$ and $\delta(q, \Phi) = q'$; or*
  - *$\chi(\Phi)$ is not satisfiable wrt $\mathcal{T}$ and $q' = q^*$; or*
  - *$q = q' = q^*$;*
- *$\mathsf{col}^{\mathcal{T}}(q^*) = 1$ and $\mathsf{col}^{\mathcal{T}}(q) = \mathsf{col}(q) + 1$, for all $q \in Q$.*

Lemma 3 *The $\mathcal{T}$-reduct of $P_{\hat{\varphi}}$ can be computed in doubly exponential time and has doubly exponential size wrt φ.*

Intuitively, the $\mathcal{T}$-reduct of $P_{\hat{\varphi}}$ accepts a trace type Ψ iff there exists a FO trace $\mathfrak{I}$ of type Ψ that does not satisfy φ and is compliant with the TBox $\mathcal{T}$, i.e. it contains only interpretations satisfying $\mathcal{T}$. Let $\mathsf{Acc}(P^{\mathcal{T}}_{\hat{\varphi}})$ be the acceptance set of $P^{\mathcal{T}}_{\hat{\varphi}}$. We have the following.

Lemma 4 *Consider a finite sequence $\Psi' = \Phi_0 \cdots \Phi_{k-1}$ of types and the finite run $\rho = q_0 \xrightarrow{\Phi_0} \cdots \xrightarrow{\Phi_{k-1}} q_k$ induced in $P_{\hat{\varphi}}$. Then, $q_k \notin \mathsf{Acc}(P^{\mathcal{T}}_{\hat{\varphi}})$ iff for every type $\Psi = \Phi_0 \cdots \Phi_{k-1}\Phi_k\Phi_{k+1}\cdots$, having Ψ' as a prefix, and for all traces $\mathfrak{I} = \mathcal{I}_0\mathcal{I}_1 \cdots$ of type Ψ, if, for all $i \geq k$, it holds that $\mathcal{I}_i \models \mathcal{T}$, then $\mathfrak{I} \models \varphi$.*

Recall that we are looking for a TKB $\Gamma' = (\mathcal{T}, \Lambda')$, obtained as a modification of $\Gamma = (\mathcal{T}, \Lambda)$, s.t. $\Gamma' \models \varphi$, i.e., all models $\mathfrak{I}$ of Γ' (Def. 7) are s.t. $\mathfrak{I} \models \varphi$ (Def. 9). Lemma 4 implies that every model $\mathfrak{I}$ of Γ' must belong to some trace type $\Psi_{\mathfrak{I}}$ whose induced run in $P_{\hat{\varphi}}$ touches some $q_k \notin \mathsf{Acc}(P^{\mathcal{T}}_{\hat{\varphi}})$, for $k = \ell' + 1$. Based on this, we next define the *repair-template* DFA.

Definition 12 (Repair-template DFA) *Given a TKB $\Gamma = (\mathcal{T}, \Lambda)$ with $\Lambda = \mathcal{A}_0 \cdots \mathcal{A}_\ell$ and a TCQ φ, let $P^{\mathcal{T}}_{\hat{\varphi}} = (2^{\mathsf{props}(\hat{\varphi})}, Q^{\mathcal{T}}, \delta^{\mathcal{T}}, q_0, \mathsf{col}^{\mathcal{T}})$ be the $\mathcal{T}$-reduct of $P_{\hat{\varphi}} = (2^{\mathsf{props}(\hat{\varphi})}, Q, \delta, q_0, \mathsf{col})$.*

The repair-template DFA (RT-DFA) for Γ and φ *is the DFA $D = (Al, S, s_0, \gamma, F)$ s.t.:*

- $Al = (\{\text{fix}, \text{add}\} \times 2^{2^{\mathsf{props}(\hat{\varphi})}}) \cup \{\text{del}\}$ *is the alphabet;*
- $S = 2^Q \times \{0, \ldots, \ell + 1\}$ *is the set of states;*
- $s_0 = (\{q_0\}, 0)$ *is the initial state;*
- $\gamma : S \times Al \to S$ *is the transition function s.t.* $\gamma((Z, i), X) = (Z', i')$ *iff either:*
  1. $X = \text{del}$, $Z = Z'$, *and* $i' = \min\{i + 1, \ell + 1\}$*; or*
  2. *all of the following hold:*
     (a) $X = (\sigma, \Upsilon)$*, with* $\sigma \in \{\text{fix}, \text{add}\}$*;*
     (b) $q' \in Z'$ *iff* $\delta(q, \Phi) = q'$*, for* $q \in Z$ *and* $\Phi \in \Upsilon$*;*
     (c) *there exists an ABox $\mathcal{A}$ consistent with $\mathcal{T}$ s.t.:* $(\mathcal{T}, \mathcal{A}) \models \bigvee_{\Phi \in \Upsilon} \chi(\Phi) \wedge \bigwedge_{\Phi \notin \Upsilon} \neg\chi(\Phi)$*;*
     (d) $i' = \begin{cases} \min\{i + 1, \ell + 1\}, & \text{if } \sigma = \text{fix} \\ i, & \text{if } \sigma = \text{add} \end{cases}$
- $F = \{(Z, \ell + 1) \in S \mid Z \cap \mathsf{Acc}(P^{\mathcal{T}}_{\hat{\varphi}}) = \emptyset\}$ *is the set of final states.*

Lemma 5 *The RT-DFA for a TKB Γ and a TCQ φ can be computed in triply exponential time and has triply exponential size wrt φ.*

Observe that the right-hand side expression of the entailment ($\models$) in Item 2c above is a Boolean combination of BCQs. The purpose of the RT-DFA is to capture the solution space of TKB Alignment for $(\mathcal{T}, \Lambda)$ and φ, in the following sense: *(i)* from every accepted word w, some TKB modification η can be derived s.t. $\mathsf{mod}(\eta, \Gamma) \models \varphi$, and *(ii)* every TKB modification η s.t. $\mathsf{mod}(\eta, \Gamma) \models \varphi$ can be derived from some accepted word w. This is formalized next, by the notion of TKB-modification *abstraction* and the subsequent result.

Definition 13 (TKB-modification abstraction) *Consider a TKB $\Gamma = (\mathcal{T}, \Lambda)$, with $\Lambda = \mathcal{A}_0 \cdots \mathcal{A}_\ell$, a TCQ φ, and let $D = (Al, S, s_0, \gamma, F)$ be the RT-DFA for Γ and φ. A word $w = w_0 \cdots w_m \in Al^*$, inducing a finite run $\rho = s_0 \xrightarrow{w_0} \cdots \xrightarrow{w_m} s_{m+1}$ in D, is an* abstraction *of (or* abstracts*) a TKB-modification $\eta = \eta_0 \cdots \eta_m$ iff, for $j = 0, \ldots, m$:*

- $w_j = \text{del}$ *and* $\eta_j = \text{del}$*; or*
- $w_j = (\text{fix}, \Upsilon)$ *and, for* $s_j = (Z, i)$, $\eta_j = (\text{fix}, \mu)$*, with* $(\mathcal{T}, \mathsf{mod}(\mu, \mathcal{A}_i)) \models \bigvee_{\Phi \in \Upsilon} \chi(\Phi) \wedge \bigwedge_{\Phi \notin \Upsilon} \neg\chi(\Phi)$*; or*
- $w_j = (\text{add}, \Upsilon)$ *and* $\eta_j = (\text{add}, \mu)$*, with* $(\mathcal{T}, \mathsf{mod}(\mu, \mathcal{A}_\emptyset)) \models \bigvee_{\Phi \in \Upsilon} \chi(\Phi) \wedge \bigwedge_{\Phi \notin \Upsilon} \neg\chi(\Phi)$.

When this holds, η is an instantiation *of (or* instantiates*) w.*

Theorem 3 *Consider a TKB $\Gamma = (\mathcal{T}, \Lambda)$, with $\Lambda = \mathcal{A}_0 \cdots \mathcal{A}_\ell$, a TCQ φ, and let $D = (Al, S, s_0, \gamma, F)$ be the RT-DFA for Γ and φ. Then:*

1. *for every word $w \in Al^*$, there exists an instantiation η s.t. $w \in \mathcal{L}(D)$ iff $\mathsf{mod}(\eta, \Gamma) \models \varphi$;*
2. *for every word $w \in Al^*$ and every instantiation η of w, it holds that $w \in \mathcal{L}(D)$ iff $\mathsf{mod}(\eta, \Gamma) \models \varphi$;*
3. *for every TKB-modification η there exists a unique abstraction w_η s.t. $w_\eta \in \mathcal{L}(D)$ iff $\mathsf{mod}(\eta, \Gamma) \models \varphi$.*

Thm. 3 states that the language of D characterizes the set of solutions for the TKB-alignment of Γ against the specification φ; in particular, Item 2 ensures that every instantiation of some TKB-modification abstraction $w \in \mathcal{L}(D)$ is a solution to TKB Alignment. Then, every optimal solution η^* is s.t.:

$$\eta^* = \text{argmin}_\eta\{\mathsf{c}(\eta, \Gamma) \mid \eta \text{ instantiates some } w \in \mathcal{L}(D)\}.$$

Based on this, we can reduce the problem of finding η^* to that of finding a minimal path in a suitably weighted graph.

Definition 14 (Minimal-instantiation Graph) *Consider a TKB $\Gamma = (\mathcal{T}, \Lambda)$, with $\Lambda = \mathcal{A}_0 \cdots \mathcal{A}_\ell$, a TCQ φ, and let $D = (Al, S, s_0, \gamma, F)$ be the RT-DFA for Γ and φ. The* minimal-instantiation graph *for Γ and φ is the weighted graph $G = (N, E, \mathsf{w})$, where:*

1. *$N = S$ is the finite set of nodes;*
2. *$E \subseteq N \times \mathsf{Atm} \times N$, is the finite set of edges, labelled by atomic TKB-modifications;*
3. *$\mathsf{w} : E \to \mathbb{R}^+$ is the edge weight function;*
4. *it holds that $e = ((Z, i), \eta, (Z', i')) \in E$ and $\mathsf{w}(e) = c$ iff, for some X, $(Z', i') = \gamma((Z, i), X)$, and:*
   - *if $X = \text{del}$ then $\eta = \text{del}$ and $c = \mathsf{c}(\text{del}, \mathcal{A}_i)$;*
   - *if $X = (\text{fix}, \Upsilon)$ then* $\eta = \text{argmin}_{(\text{fix},\mu)} \{\mathsf{c}((\text{fix}, \mu), \mathcal{A}_i) \mid (\mathcal{T}, \mathsf{mod}(\mu, \mathcal{A}_i)) \models \bigvee_{\Phi \in \Upsilon} \chi(\Phi) \wedge \bigwedge_{\Phi \notin \Upsilon} \neg\chi(\Phi)\}$ *and* $c = \mathsf{c}(\eta, \mathcal{A}_i)$*;*
   - *if $X = (\text{add}, \Upsilon)$ then* $\eta = \text{argmin}_{(\text{add},\mu)} \{\mathsf{c}((\text{add}, \mu), \Lambda_\epsilon) \mid (\mathcal{T}, \mathsf{mod}(\mu, \mathcal{A}_\emptyset)) \models \bigvee_{\Phi \in \Upsilon} \chi(\Phi) \wedge \bigwedge_{\Phi \notin \Upsilon} \neg\chi(\Phi)\}$ *and* $c = \mathsf{c}(\eta, \mathcal{A}_\emptyset)$.

Lemma 6 *The Minimal-instantiation Graph for a TKB Γ and a TCQ φ can be computed in triply exponential time and has triply exponential size wrt φ.*

The minimal-instantiation graph G is a graph whose edges are labelled with atomic TKB-modifications and weighted with the corresponding cost. Through its labels, every finite path $s_0 \xrightarrow{\eta_0} \cdots \xrightarrow{\eta_{m-1}} s_m$ of G defines an instantiation $\eta = \eta_0 \cdots \eta_m$ of some (not necessarily accepted) input word $w = w_0 \cdots w_m$ of D. Also the viceversa holds, i.e., every input word w of D is an abstraction of the TKB-modification η defined by some path of G.

Observe that, by Item 4 of Def. 14, η includes only minimal-cost atomic TKB-modifications η_i, thus it is a minimal-cost TKB-modification among all those that instantiate the same w. Moreover, recall that, by Theorem 3, every solution to TKB Alignment is associated to an abstraction $w \in \mathcal{L}(D)$. Thus, since every such w has a minimal-cost instantiation in some path of G, we can search for the minimal-cost solution by exploring the paths of G. Indeed, it is enough to search for the minimal-cost paths of G which correspond to the words w accepted by D; since the nodes of G correspond to the states of D, this corresponds to searching for a minimal-path of G starting in s_0 and ending in some node that is an accepting state for D.

Theorem 4 *Consider a TKB $\Gamma = (\mathcal{T}, \Lambda)$, with $\Lambda = \mathcal{A}_0 \cdots \mathcal{A}_\ell$, a TCQ φ, and let $D = (Al, S, s_0, \gamma, F)$ be the RT-DFA for Γ and φ. A TKB-modification $\eta = \eta_0 \cdots \eta_m$ is an optimal solution of TKB Alignment for Γ and φ iff there exists a minimum-cost path $\pi = n_0 \xrightarrow{\eta_0} \cdots \xrightarrow{\eta_{m-1}} n_m$ in G, s.t. $n_0 = s_0$ and $n_m \in F$.*

Thus, with the minimal-instantiation graph G at hand, the search can be easily performed using, e.g., Dijkstra's algorithm. Observe however that, in order to obtain effective solvability of TKB Alignment, it remains to show that G is actually computable. In particular, it remains to explain how to obtain the labels and the weights of G.

By Def. 14 (Item 4), computing the labels and the weights of G requires to solve, for every edge, one *local minimization problem* of the form $\mathrm{argmin}_{(\sigma,\mu)}\{\mathsf{c}((\sigma,\mu),\Lambda)\}$, subject to a constraint of the form $(\mathcal{T}, \mathsf{mod}(\mu, \mathcal{A})) \models \phi$, with $\phi \in \mathcal{B}(\textsc{bcq})$. This is the *KB-alignment* problem, which we formally define and solve in Section 4. We report here that the problem can be solved in doubly exponential time. This, together with the complexity results reported above, leads to the following.

Theorem 5 *TKB Alignment is solvable in triply exponential time.*

As mentioned above, one might consider the semantics on TCQs given in [4] and instantiate TKB alignment with it. Such variation on the semantics has minimal impact on the solution technique, which can be adapted to that setting with minimal changes. Details are omitted, due to space constraints.

4 The KB-Alignment Problem

In this section, we define the *knowledge base alignment problem (KB Alignment)*, and show how to solve it for the query language $\mathcal{B}(\textsc{bcq})$ and knowledge bases written in the DL $\mathcal{ALC}$.

Definition 15 (KB Alignment) *Given a KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ and a query $\phi \in \mathcal{B}(\textsc{bcq})$, the problem of KB Alignment consists in finding an ABox-modification μ such that for $\mathcal{A}' = \mathsf{mod}(\mu, \mathcal{A})$:*

1. *$(\mathcal{T}, \mathcal{A}')$ is consistent;*
2. *$(\mathcal{T}, \mathcal{A}') \models \phi$; and*
3. *$\mathsf{c}(\mu) \leq \mathsf{c}(\mu')$ for all ABox-modifications μ' satisfying Conditions 1 and 2.*

An ABox-modification μ satisfying Condition 1 and 2 is a solution to KB Alignment *for $(\mathcal{K}, \phi)$; if it also satisfies 3, it is an* optimal solution.

To solve an instance of KB Alignment, it is sufficient to consider ABox-modifications with operations defined using concept and role names from the input KB and query, i.e., using names from their signature. A *signature* is a finite subset of $\mathsf{N_C} \cup \mathsf{N_R}$. We use $\mathsf{sig}(X)$ to denote the set of concept and role names occurring in X, where X can be a KB or a query.

Lemma 7 *Let $\mathcal{K}$ be a KB and $\phi \in \mathcal{B}(\textsc{bcq})$. If KB Alignment has a solution for $(\mathcal{K}, \phi)$, then it has an optimal solution μ where all ABox-operations $\mathrm{ins}(\alpha)$ and $\mathrm{rem}(\alpha)$ in μ are such that α is an assertion defined over $\mathsf{sig}(\mathcal{K}) \cup \mathsf{sig}(\phi)$.*

An algorithm that solves KB Alignment for $\mathcal{ALC}$ KBs and queries in $\mathcal{B}(\textsc{bcq})$ is introduced next.

4.1 Solving KB-Alignment

Algorithm 1 describes our method to solve KB Alignment. The approach is to (1) compute an upper bound on the cost of optimal solutions (if any exists), (2) iterate over all ABox-modifications of decreasing cost starting from the upper bound, and finally output a modification that satisfies Condition 1 to 3 from Definition 15, i.e., which is optimal.

In order to obtain the upper bound on the ABox-modification the algorithm first computes for the input TBox $\mathcal{T}$ and query ϕ, an ABox $\mathcal{A}'$ such that $(\mathcal{T}, \mathcal{A}')$ is consistent and $(\mathcal{T}, \mathcal{A}') \models \phi$. Then, a modification μ^* from the input ABox $\mathcal{A}$ into $\mathcal{A}'$ is obtained. Such a modification always exists: given two ABoxes $\mathcal{A}$ and $\mathcal{A}'$, a *trivial modification from $\mathcal{A}$ into $\mathcal{A}'$* is defined as a sequence $\mu = m_1 \cdots m_k \cdots m_n$, where $m_1 \cdots m_k$ consists of the removal of all assertions in $\mathcal{A}$ and $m_{k+1} \cdots m_n$ consists of the insertion of all assertions present in $\mathcal{A}'$.

The algorithm computes one trivial modification, which, by Definition 15, is a (possibly non-optimal) solution, thus $\mathsf{c}(\mu^*)$ realizes the first step of the approach as it is an upper bound on the cost of optimal solutions. Then, the *for-loop* enumerates all ABox-modifications with cost smaller than $\mathsf{c}(\mu^*)$, that satisfy Condition 1 and 2 of Definition 15, and it returns one ABox modification of minimal cost.

By using Lemma 7, it is not hard to show that the output of Algorithm 1 is always an optimal solution.

Lemma 8 *If Algorithm 1 returns μ^* on input $\mathcal{K}$ and ϕ, then μ^* is an optimal solution to KB Alignment for $(\mathcal{K}, \phi)$. If Algorithm 1 returns "no solution", then KB Alignment has no solution for $(\mathcal{K}, \phi)$.*

Hence, to see that Algorithm 1 solves the KB-alignment problem, it remains to show that it terminates, i.e., all its steps can *effectively* be computed. The following arguments show this for most of the steps:

- Since $\mathcal{A}$ and $\mathcal{A}'$ are finite sets, a trivial modification μ^* can easily be computed from the ABox $\mathcal{A}'$.
- Consistency of $\mathcal{ALC}$ KBs is a decidable problem ([5]), as well as entailment of $\mathcal{B}(\textsc{bcq})$-queries in $\mathcal{ALC}$ (see Sec. 2.2), hence the conditions at line 5 can be effectively verified for each $\mu \in M$.
- The set M contains only ABox-modifications defined over the finite signature $\mathsf{sig}(\mathcal{K}) \cup \mathsf{sig}(\phi)$. Hence, given $n > 0$, M contains finitely many μ with n ABox-operations, and each such μ has cost of at least $c \cdot n$, where c is the minimal cost of an ABox operation defined over $\mathsf{sig}(\mathcal{K}) \cup \mathsf{sig}(\phi)$. This implies that M is a finite set and contains only modifications with no more than $\mathsf{c}(\mu^*)/c$ operations. Thus, M can be visited in finite time.

Algorithm 1 KB Alignment.

Input: An $\mathcal{ALC}$ KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ and a query $\phi \in \mathcal{B}(\text{BCQ})$.
Output: An optimal solution of KB Alignment for $(\mathcal{K}, \phi)$, if a solution exists; or "no solution", otherwise.

```
1: Compute an ABox $\mathcal{A}'$ s.t. $(\mathcal{T}, \mathcal{A}')$ is consistent, $(\mathcal{T}, \mathcal{A}') \models \phi$
   and $\mathsf{sig}(\mathcal{A}') \subseteq \mathsf{sig}(\mathcal{K}) \cup \mathsf{sig}(\phi)$. If no such ABox exists, return
   "no solution";
2: Define a trivial modification $\mu^*$ from $\mathcal{A}$ into $\mathcal{A}'$;
3: Let $M$ be the set of ABox-modifications $\mu$ defined over $\mathsf{sig}(\mathcal{K}) \cup$
   $\mathsf{sig}(\phi)$ s.t. $\mathsf{c}(\mu) < \mathsf{c}(\mu^*)$;
4: for all $\mu \in M$ do
5:     if ($\mu$ satisfies conditions 1 and 2 in Definition 15) and
           $(\mathsf{c}(\mu) < \mathsf{c}(\mu^*))$ then
6:             $\mu^* := \mu$;
7: return $\mu^*$;
```

It remains to specify how to compute the initial ABox $\mathcal{A}'$ (or to determine that it does not exist). This requires a more involved argument presented in the following.

4.1.1 *Computing the initial ABox $\mathcal{A}'$*

The computation of the initial ABox $\mathcal{A}'$ in our algorithm is closely related to the *query emptiness problem* in ontology-mediated query answering. This problem was introduced and investigated in [2] for various DLs (including $\mathcal{ALC}$) and the query language CQ. We define this problem here for the more general query language $\mathcal{B}(\text{CQ})$. It uses the notion of Σ-ABoxes, which refers to ABoxes that use only names from a signature Σ.

Definition 16 (**$\mathcal{B}$(CQ)-query Emptiness**) *Let $\mathcal{T}$ be a TBox, Σ a signature and $\phi \in \mathcal{B}(\text{CQ})$. The query ϕ is called* empty for Σ given $\mathcal{T}$ *if for all Σ-ABoxes $\mathcal{A}$ such that $(\mathcal{T}, \mathcal{A})$ is consistent, we have $cert_{(\mathcal{T},\mathcal{A})}(\phi) = \emptyset$.*

$\mathcal{B}$(CQ)-query Emptiness *is the problem of deciding, given a TBox $\mathcal{T}$, a signature Σ, and $\phi \in \mathcal{B}(\text{CQ})$, whether ϕ is empty for Σ wrt $\mathcal{T}$.*

In [2], it is shown that to decide CQ-query Emptiness in $\mathcal{ALC}$, it suffices to consider a single Σ-ABox $\mathcal{A}_{\mathcal{T},\Sigma}$. This ABox is of exponential size and can be computed (from a given satisfiable TBox $\mathcal{T}$ and a signature Σ) in exponential time, in the size of $\mathcal{T}$ and the cardinality of Σ. Moreover, it satisfies the following properties:

- the KB $(\mathcal{T}, \mathcal{A}_{\mathcal{T},\Sigma})$ is consistent, and
- given a pure CQ ϕ, ϕ is empty for Σ wrt $\mathcal{T}$ iff $cert_{(\mathcal{T},\mathcal{A}_{\mathcal{T},\Sigma})}(\phi) = \emptyset$.

The arguments used to prove the second property can be easily extended to pure $\mathcal{B}(\text{CQ})$-queries, as the following result shows.

Lemma 9 *Let $\mathcal{T}$ be a satisfiable $\mathcal{ALC}$ TBox, Σ a signature and $\phi \in \mathcal{B}(\text{CQ})$ a pure query. Then, ϕ is empty for Σ wrt $\mathcal{T}$ iff $cert_{(\mathcal{T},\mathcal{A}_{\mathcal{T},\Sigma})}(\phi) = \emptyset$.*

For *non-pure* queries, query emptiness can be reduced to the case of pure queries. This can be done as follows. Let $\mathcal{T}$ be an $\mathcal{ALC}$ TBox, Σ a signature, and $\phi \in \mathcal{B}(\text{CQ})$ a *non-pure* query with $\mathsf{Ind}(\phi) = \{a_1, \ldots, a_m\}$. We select m *fresh* concept names $A_1, \ldots, A_m$, i.e., concept names neither occurring in $\mathcal{T}, \phi$ nor Σ. Then, we define an $\mathcal{ALC}$ TBox $\mathcal{T}_p$, a signature Σ_p and a query $\phi_p \in \mathcal{B}(\text{CQ})$, as follows:

- $\mathcal{T}_p = \mathcal{T} \cup \mathcal{T}_d$, where $\mathcal{T}_d = \{A_i \sqcap A_j \sqsubseteq \bot \mid 1 \leq i < j \leq m\}$,
- $\phi_p = \phi_x \wedge \phi_d$, where ϕ_x is obtained from ϕ by replacing each $a \in \mathsf{Ind}(\phi)$ by a fresh free variable x_a, whereas ϕ_d is the CQ $\phi_d = A_1(x_{a_1}) \wedge \ldots \wedge A_m(x_{a_m})$, and
- $\Sigma_p = \Sigma \cup \{A_1, \ldots, A_m\}$.

The following lemma shows that testing whether ϕ is empty for Σ wrt $\mathcal{T}$ reduces to checking emptiness of ϕ_p for Σ_p wrt $\mathcal{T}_p$. Since ϕ_p is a pure query, this yields a reduction from query emptiness of *non-pure* queries to the case of pure queries.

Lemma 10 *Let $\mathcal{T}$ be an $\mathcal{ALC}$ TBox, Σ a signature, and ϕ a query in $\mathcal{B}(\text{CQ})$. The following holds:*

1. *ϕ is empty for Σ wrt $\mathcal{T}$ iff ϕ_p is empty for Σ_p wrt $\mathcal{T}_p$.*
2. *If there is a Σ_p-ABox $\mathcal{A}_p$ that witnesses non-emptiness of ϕ_p for Σ_p wrt $\mathcal{T}_p$, then, given $t \in cert_{(\mathcal{T}_p,\mathcal{A}_p)}(\phi_p)$, $\mathcal{A}_p$ can be transformed in polynomial time (in the size of $\mathcal{A}_p$ and t) into a Σ-ABox $\mathcal{A}$ witnessing non-emptiness of ϕ for Σ wrt $\mathcal{T}$.*

Hence, if the input query ϕ of Algorithm 1 is pure, by Lemma 9 the search space for the ABox $\mathcal{A}'$ can simply be restricted to $\{\mathcal{A}_{\mathcal{T},\Sigma}\}$ where $\Sigma = \mathsf{sig}(\mathcal{K}) \cup \mathsf{sig}(\phi)$. Otherwise, Lemma 10 tells us how to obtain $\mathcal{A}'$ (if it exists). Namely, the algorithm first constructs $\mathcal{T}_p, \phi_p$ and Σ_p from $\mathcal{T}, \phi$ and Σ. It then checks whether ϕ_p is non-empty for Σ_p wrt $\mathcal{T}_p$, by using $\mathcal{A}_{\mathcal{T}_p,\Sigma_p}$. If the latter is true, then $\mathcal{A}'$ is selected as the ABox obtained from applying to $\mathcal{A}_{\mathcal{T}_p,\Sigma_p}$ the transformation from the second statement in Lemma 10. Overall, this provides a way to compute $\mathcal{A}'$ whenever it exists.

Hence, Algorithm 1 always terminates. This, together with Lemma 8, yields solvability of KB Alignment. A closer look at Algorithm 1 reveals that it runs in *double exponential time* in the size of the input KB and query. Thus, we obtain the following result.

Theorem 6 *KB Alignment is solvable for $\mathcal{ALC}$ and $\mathcal{B}(\text{BCQ})$ in* double exponential time.

5 Discussion and Future Work

TKB Alignment is a new variant of the alignment problem that admits richer state and property descriptions by DL knowledge bases. Our setting uses TKBs written in $\mathcal{ALC}$, CQs with LTL operators, and a cost function for the edit operations. We have shown that TKB Alignment wrt temporal CQs is effectively solvable, by developing computation methods for both TKB and KB Alignment.

The TKB-alignment problem is closely related to abduction and to computing repairs of KBs, as these tasks also change a KB to either gain a desired consequence or remove an unwanted one. However, although being active research topics, neither of the two has yet been intensively investigated for the temporalized setting and entailment of TCQs. Furthermore, TKB Alignment requires a cost-optimal solution, which is not a common setting covered in research dedicated to abduction or repairs.

Interestingly, the task of TKB Alignment can also be used for computing relaxed answers of temporal CQs in the following way. Given a tuple of individuals $\bar{a}$ which is not a certain answer of a given TCQ ϕ, solve TKB Alignment for the Boolean TCQ obtained from ϕ' by assigning $\bar{a}$ to the answer variables of ϕ. The costs computed during TKB Alignment for ϕ' are then a measure of the "distance" to a certain answer of the query. Relaxed answers are then those tuples of individuals with costs below a given threshold.

Our initial investigation on TKB Alignment uses a unitary cost measure for the edit operations mostly to ease presentation, as our approach can handle other cost measures easily. In this work, we did not regard rigid symbols, which are left for future work.

Acknowledgements

The work of Giuseppe Perelli was partially funded by MUR under the PRIN programme, grant B87G22000450001 (PINPOINT), and by the PNRR MUR project PE0000013-FAIR. The work of Fabio Patrizi was partially funded by MUR under the PNRR MUR project PE0000013-FAIR, the ERC Advanced Grant WhiteMech (No. 834228), and the Sapienza Project MARLeN. The work of Oliver Fernández Gil and Anni-Yasmin Turhan was partially funded by the AI competence center ScaDS.AI Dresden/Leipzig.

References

[1] Alessandro Artale, Roman Kontchakov, Alisa Kovtunova, Vladislav Ryzhikov, Frank Wolter, and Michael Zakharyaschev, 'First-order rewritability of ontology-mediated queries in linear temporal logic', *Artif. Intell.*, **299**, 103536, (2021).

[2] Franz Baader, Meghyn Bienvenu, Carsten Lutz, and Frank Wolter, 'Query and predicate emptiness in ontology-based data access', *J. Artif. Intell. Res.*, **56**, 1–59, (2016).

[3] Franz Baader, Stefan Borgwardt, Patrick Koopmann, Veronika Thost, and Anni-Yasmin Turhan, 'Semantic technologies for situation awareness', *Künstliche Intell.*, **34**(4), 543–550, (2020).

[4] Franz Baader, Stefan Borgwardt, and Marcel Lippmann, 'Temporal query entailment in the description logic SHQ', *J. Web Semant.*, **33**, 71–93, (2015).

[5] *The Description Logic Handbook: Theory, Implementation, and Applications*, eds., Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, Cambridge University Press, 2003.

[6] Meghyn Bienvenu, 'A short survey on inconsistency handling in ontology-mediated query answering', *Künstliche Intell.*, **34**(4), 443–451, (2020).

[7] Meghyn Bienvenu and Magdalena Ortiz, 'Ontology-mediated query answering with data-tractable description logics', in *Reasoning Web. Web Logic Rules - 11th International Summer School*, volume 9203 of *Lecture Notes in Computer Science*, pp. 218–307. Springer, (2015).

[8] Stefan Borgwardt, Marcel Lippmann, and Veronika Thost, 'Temporalizing rewritable query languages over knowledge bases', *Journal of Web Semantics*, **33**, 50–70, (2015).

[9] Camille Bourgaux, Patrick Koopmann, and Anni-Yasmin Turhan, 'Ontology-mediated query answering over temporal and inconsistent data', *Semantic Web*, **10**(3), 475–521, (2019).

[10] Diego Calvanese, Magdalena Ortiz, Mantas Simkus, and Giorgio Stefanoni, 'The complexity of conjunctive query abduction in DL-Lite', in *Proceedings of the 24th International Workshop on Description Logics (DL 2011)*, volume 745 of *CEUR Workshop Proceedings*. CEUR-WS.org, (2011).

[11] Giuseppe De Giacomo, Fabrizio Maria Maggi, Andrea Marrella, and Fabio Patrizi, 'On the disruptive effectiveness of automated planning for LTL_f-based trace alignment', in *AAAI*, pp. 3555–3561, (2017).

[12] Massimiliano de Leoni, Fabrizio Maria Maggi, and Wil van der Aalst, 'An alignment-based framework to check the conformance of declarative process models and to preprocess event-log data', *Inf. Syst.*, **47**, 258–277, (2015).

[13] Massimiliano de Leoni, Fabrizio Maria Maggi, and Wil M. P. van der Aalst, 'Aligning event logs and declarative process models for conformance checking', in *BPM*, (2012).

[14] Warren Del-Pinto and Renate A. Schmidt, 'ABox abduction via forgetting in $\mathcal{ALC}$', in *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019*, pp. 2768–2775. AAAI Press, (2019).

[15] Oliver Fernandez-Gil, Fabio Patrizi, Giuseppe Perelli, and Anni-Yasmin Turhan. Optimal alignment of temporal knowledge bases. CoRR abs/2307.15439, 2023.

[16] Dov M. Gabbay, 'The declarative past and imperative future: Executable temporal logic for interactive systems', in *Temporal Logic in Specification, Altrincham, UK, Proceedings*, volume 398 of *Lecture Notes in Computer Science*, pp. 409–448. Springer, (1987).

[17] Valerie King, Orna Kupferman, and Moshe Y. Vardi, 'On the complexity of parity word automata', in *Proc. of FOSSACS*, eds., Furio Honsell and Marino Miculan, volume 2030 of *Lecture Notes in Computer Science*, pp. 276–286. Springer, (2001).

[18] Patrick Koopmann, Warren Del-Pinto, Sophie Tourret, and Renate Schmidt, 'Signature-based abduction for expressive description logics', in *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning (KR 2020)*, (2020).

[19] Orna Kupferman, Nir Piterman, and Moshe Y. Vardi, 'Extended temporal logic revisited', in *CONCUR 2001 - Concurrency Theory, 12th International Conference, Aalborg, Denmark, Proceedings*, eds., Kim Guldstrand Larsen and Mogens Nielsen, volume 2154 of *Lecture Notes in Computer Science*, pp. 519–535. Springer, (2001).

[20] Carsten Lutz, 'The complexity of conjunctive query answering in expressive description logics', in *Automated Reasoning, 4th International Joint Conference, IJCAR 2008, Proceedings*, volume 5195 of *Lecture Notes in Computer Science*, pp. 179–193. Springer, (2008).

[21] Magdalena Ortiz, Mantas Simkus, and Thomas Eiter, 'Worst-case optimal conjunctive query answering for an expressive description logic without inverses', in *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008*, pp. 504–510. AAAI Press, (2008).

[22] Nir Piterman, 'From nondeterministic Büchi and Streett automata to deterministic parity automata', *Log. Methods Comput. Sci.*, **3**(3), (2007).

[23] Manfred Schmidt-Schauß and Gert Smolka, 'Attributive concept descriptions with complements', *Artif. Intell.*, **48**(1), 1–26, (1991).

[24] Ahmet Soylu, Martin Giese, Rudolf Schlatte, Ernesto Jiménez-Ruiz, Evgeny Kharlamov, Özgür L. Özçep, Christian Neuenstadt, and Sebastian Brandt, 'Querying industrial stream-temporal data: An ontology-based visual approach', *J. Ambient Intell. Smart Environ.*, **9**(1), 77–95, (2017).

[25] Moshe Y. Vardi, 'An automata-theoretic approach to linear temporal logic', in *Logics for Concurrency - Structure versus Automata (8th Banff Higher Order Workshop, Banff, Canada, Proceedings)*, volume 1043 of *Lecture Notes in Computer Science*, pp. 238–266. Springer, (1995).

[26] Thomas Wilke, 'Classifying discrete temporal properties', in *STACS 99, 16th Annual Symposium on Theoretical Aspects of Computer Science, Trier, Germany, Proceedings*, eds., Christoph Meinel and Sophie Tison, volume 1563 of *Lecture Notes in Computer Science*, pp. 32–46. Springer, (1999).