Answering Temporal Conjunctive Queries over Description Logic Ontologies for Situation Recognition in Complex Operational Domains*

Lukas Westhofen[1(✉)] , Christian Neurohr[1] , Jean Christoph Jung[2] ,
and Daniel Neider[2,3]

[1] German Aerospace Center (DLR) e.V., Institute of Systems Engineering for Future
Mobility, Oldenburg, Germany
`lukas.westhofen@dlr.de`, `christian.neurohr@dlr.de`
[2] TU Dortmund University, Dortmund, Germany
`jean.jung@tu-dortmund.de`,`daniel.neider@tu-dortmund.de`
[3] Center for Trustworthy Data Science and Security, University Alliance Ruhr,
Dortmund, Germany

Abstract. For developing safe automated systems, recognizing safety-critical situations in data from their complex operational domain is imperative. This capability is, for example, essential when evaluating the system's conformance to specified requirements in test run data. The requirements involve a temporal dimension, as the system operates over time. Moreover, the generated data are usually relational and require additional background knowledge about the domain for correctly recognizing the situation. This fact makes propositional temporal logics, an established tool, unsuitable for the task. We address this issue by developing a tailored temporal logic to query for situations in relational data over complex domains. Our language combines mission-time linear temporal logic with conjunctive queries to access time-stamped data with background knowledge formulated in an expressive description logic. Currently, however, no tools exist for answering queries in such settings. We hence also contribute an implementation in the logic reasoner OPEN-LLET, leveraging the efficacy of well-established conjunctive query answering. Moreover, we present a benchmark generator in the setting of automated driving and demonstrate that our tool performs well when tasked with recognizing safety-critical situations in road traffic.

Keywords: Temporal Conjunctive Queries · Description Logics · Temporal Logics.

---

1   Introduction

Recent technological advances in, e.g., sensors and computer vision, gave up-draft to the development of automated systems performing safety-critical tasks in complex domains. These systems are expected to safely operate without human intervention in these contexts. Consider, for example, automated driving systems (ADSs), where the responsibility of safely navigating the environment lies fully with the system [35]. The combination of their safety-critical nature and the complex operational domain makes it hard to guarantee the absence of unreasonable risk before public release, which is, however, required by many homologation authorities. Alas, correct-by-design techniques are rendered inapplicable by the high system complexity. Thus, manufactures must resort to empirically assessing the system's risk prior to deployment. As automated systems interact with their environment over time, a promising approach for risk assessment is to decompose the complex operational domain into finite-time sequences ('scenarios') [34]. Safety requirements – aiming to mitigate unreasonable risks – are then specified for these scenarios. Hence, a formal specification of the actors' temporal behavior becomes essential. An exemplary requirement reads as follows: 'In situations where the absence of pedestrians is not guaranteed, adapt the speed appropriately.' Note that this rule consists of a premise (the situation) and a consequence (the behavior). The number of situations to write requirements for can be enormous, e.g., *occlusions* [42], violating the *safety distance* [43], and maneuvers such as *passing parking vehicles* [11]. Due to their large number, testing the most widely used option for verification, i.e., to check the system's conformance with requirements. For this, data of test runs of the system operating within its environment are recorded. Adherence to the requirements is then evaluated by recognizing the situation ('no guaranteed absence of pedestrians') and testing for the implied behavior ('adapted speed'). We argue that this approach has three requirements:

Relational and Temporal Domain Formally modelling traffic situations inherently requires a relational language since they refer to individuals and their relationships, e.g., `drives`. Moreover, the number of individuals is not fixed beforehand. Finally, scenarios over such situations involve the description of temporal aspects. A typical example is the process of overtaking.

Rich Background Knowledge We do not assume that the data is complete in the sense that we can observe all facts about all individuals. Instead, we assume to have rich knowledge about the relations used in the situation descriptions. Examples for this are:
  – a `Driver` is equivalent to a `Human` which `drives` some `Vehicle`, or
  – a `Driver` is never a `Pedestrian`.
Such knowledge must be included since otherwise situations may not be correctly recognized in the data and test evaluation produces false results.

Formal Specifications of Properties It is established that specifying and testing requirements benefits greatly from formal approaches. Standard requirement formalization languages, like linear temporal logic, are however propositional and thus unsuitable for our purposes.

An established way to address the first two aspects is to model situations via *temporal knowledge bases* $\mathcal{K} = (\mathcal{O}, \mathcal{D})$ which consist of a domain ontology $\mathcal{O}$ that describes the background knowledge and a temporal database $\mathcal{D}$ that describes the evolution of the situation over time. Formally, $\mathcal{D}$ is a sequence $\mathcal{D} = (\mathcal{D}_0, \dots, \mathcal{D}_n)$ of time-stamped databases. Note that, in using temporal knowledge bases, we adopt the *open world assumption* (OWA), which intuitively says that the true facts are not only those in $\mathcal{D}$ but those that are *entailed* by $\mathcal{O}$ and $\mathcal{D}$.

As to address the third aspect above, i.e., to formally specify properties, we use a suitable extension of linear temporal logic (LTL). Recall that LTL is a language for describing properties over a set of propositions by using modalities such as $\Diamond\varphi$ (φ holds *eventually*), $\Box\varphi$ (φ holds *globally*), $\varphi_1 \mathcal{U} \varphi_2$ (φ_1 holds *until* φ_2), and $\bigcirc\varphi$ (φ holds in the *next* step). Unfortunately, this does not suffice when working over relational data. A natural option to extend LTL in the required way is to replace propositions by *queries*. In this work, we use conjunctive queries (CQs). CQs are one of the most common query language for databases and expressively equivalent to the SELECT-FROM-WHERE fragment of SQL. For example, we can ask for all drivers d of a vehicle by the CQ $\exists v.\mathtt{Vehicle}(v) \wedge \mathtt{drives}(d, v)$ with one existentially quantified variable v and one answer variable d. In terms of the temporal expressivity, our application further requires that

(1) we operate on finite traces whose length is bounded by the length of the temporal database $\mathcal{D}$ specified in the temporal knowledge base,
(2) as duration constraints are used in specifications, e.g., to distinguish maneuvers of certain lengths, we incorporate metric operators, and
(3) we analyze the data a-posteriori. Hence we are not in a run-time verification setting and require only future time operators.

We term the resulting language *metric temporal conjunctive queries (MTCQs)*, which features both unbounded and bounded future time operators over finite traces and uses CQs in its atoms and is based on Mission-Time LTL (MLTL) [29]. MTCQs can, for example, express properties like $\Phi_0^{ex}(x) = \Diamond\neg\mathtt{Pedestrian}(x)$, asking for all individuals x that are eventually not a pedestrian. A more involved MTCQ asking for all x that move past a parking vehicle y on a two-lane road is

$$\Phi_1^{ex}(x, y) = \Box\big(\exists r.\mathtt{Vehicle}(x) \wedge \mathtt{2\_Lane\_Road}(r) \wedge \mathtt{intersects}(r, x) \wedge$$
$$\mathtt{Parking\_Vehicle}(y)\big) \wedge \Diamond\big(\mathtt{in\_front\_of}(y, x) \wedge \bigcirc$$
$$\big((\mathtt{in\_proximity}(x, y) \wedge \mathtt{to\_the\_side\_of}(y, x)) \mathcal{U} \mathtt{behind}(y, x)\big)\big).$$

Recognizing such a situation for checking a requirement translates to the task of evaluating an MTCQ $\Phi(\vec{x})$ with answer variables $\vec{x}$ over a temporal knowledge base $\mathcal{K}$. Informally, if we want to verify that a tuple of individuals $\vec{a}$ conforms to some specification $\Phi(\vec{x})$ in a situation $\mathcal{K} = (\mathcal{O}, \mathcal{D})$, we have to check whether the entailment $(\mathcal{O}, \mathcal{D}) \models \Phi(\vec{a})$ is true, cf. Section 3 for precise definitions.

This task obviously depends on the chosen ontology language. For this, we use description logics (DLs), an established knowledge representation formalism, which offers a good compromise between complexity and expressivity [10]. Our

approach works up to the $\mathcal{SRIQ}^{(\mathcal{D})}$ fragment of DLs. It is close to the formalism behind the Web Ontology Language (OWL) 2, an expressive and widespread DL language. The mentioned task of entailment has been studied for DL temporal knowledge bases and a different yet related extension of LTL [8], cf. Section 2.

We now illustrate this setup by means of a simple example. A DL ontology $\mathcal{O}$ is a set of concept inclusions $C \sqsubseteq D$ for concept descriptions C and D. We also write $C \equiv D$ to denote concept equivalence. DLs allow for arbitrary names as basic concepts. We have special names for nothing ($\bot$) and all things ($\top$). Besides concepts, DLs also allow so-called roles (relations) between concepts. From these, we can inductively build new concepts. For an example ontology $\mathcal{O}^{ex}$, we might state that every driver is a human by $\texttt{Driver} \sqsubseteq \texttt{Human} \in \mathcal{O}^{ex}$. As to illustrate the combination of roles and concepts we define drivers as the intersection (using the $\sqcap$-operator) of all humans and all things that drive some (using the $\exists$-operator) vehicle, written as $\texttt{Driver} \equiv \texttt{Human} \sqcap \exists \texttt{drives.Vehicle} \in \mathcal{O}^{ex}$. We can use $\bot$ to express that pedestrians and drivers are disjoint: $\texttt{Driver} \sqcap \texttt{Pedestrian} \sqsubseteq \bot \in \mathcal{O}^{ex}$.

These operators may be enough for simple domains. However, knowledge about relations in complex domains is often involved, in which case even more expressive operators can be allowed. For example, the MTCQ Φ_1^{ex} requires recognizing situations of passing parking vehicles. Here, expressive DLs allow modeling two-lane roads to have exactly two lanes (by the concept $=2\,\texttt{has\_lane.Lane}$) and be a road (by the concept $\texttt{Road} \sqcap =2\,\texttt{has\_lane.Lane}$). Moreover, parking vehicles are standing (with a speed of the datatype literal 0.0) dynamical objects on a parking spot. This is expressed by the following DL ontology:

- $\texttt{2\_Lane\_Road} \equiv \texttt{Road} \sqcap =2\,\texttt{has\_lane.Lane}$
- $\texttt{Vehicle} \sqcap \texttt{Standing\_Dynamical\_Object} \sqcap \exists \texttt{intersects.Parking\_Spot} \sqsubseteq$
  $\texttt{Parking\_Vehicle}$
- $\texttt{Parking\_Spot} \equiv \texttt{Parking\_Lane} \sqcup \texttt{Walkway}$
- $\texttt{Standing\_Dynamical\_Object} \equiv \texttt{Dynamical\_Object} \sqcap \exists \texttt{has\_speed.}\{0.0\}$

Let us now use the simple example to give an intuition on the semantics of MTCQs over DL ontologies. First, we create an exemplary database with facts over so-called individuals (concrete objects that are perceived). For example, we can assert for the first time point that the individual $\texttt{h}$ is a human driving the individual $\texttt{v}$, a vehicle, by writing the facts as $\mathcal{D}_0^{ex} = \{\texttt{Human(h)}, \texttt{drives(h, v)}, \texttt{Vehicle(v)}\}$. Next, we may perceive $\mathcal{D}_1^{ex} = \emptyset$, i.e., no information at all. Together with the ontology, it forms a temporal knowledge base $\mathcal{K}^{ex} = (\mathcal{O}^{ex}, (\mathcal{D}_0^{ex}, \mathcal{D}_1^{ex}))$. If we query $\mathcal{K}^{ex}$ w.r.t. $\Phi_0^{ex}(x) = \Diamond \neg \texttt{Pedestrian}(x)$, we get $\texttt{h}$ as the only answer, as $\texttt{h}$ is a driver in $\mathcal{D}_0^{ex}$ and the ontology states that drivers can never be pedestrians. However, if we change the query to $\Phi_2^{ex} = \Box \neg \texttt{Pedestrian}(x)$, no individual satisfies the constraint, since $\mathcal{D}_1^{ex}$ asserts nothing – it can very well be possible that $\texttt{h}$ has become a pedestrian (due to the OWA).

This example highlights that languages like MTCQs are important for testing requirements on systems in complex domains. However, up to now, only the theoretical work by Baader et al. examines a related but hard-to-implement setting over infinite traces for complexity-theoretic analyses [8]. No language

has yet been defined that is practically suitable for implementation and has the required expressiveness. Moreover, there currently is no tooling for *any* temporal query language over expressive DLs. Our work on MTCQs addresses this gap.

For this, we first introduce the formal foundation of MTCQs in Section 3. We implement the framework in an answering engine for a large and practically relevant subclass of MTCQs in Section 4, closing the identified research gap. To evaluate its efficacy, we present a benchmark generator for temporal knowledge bases, as described in Section 5. We show the efficacy of our tool in this practical setting in Section 6. To summarize, the main contributions of our work are

1. MTCQs as a practically implementable and expressive temporal query language and the first tool for answering such queries up to the DL $\mathcal{SRIQ}^{(\mathcal{D})}$,
2. a benchmark generator for the evaluation of inference tasks on temporal knowledge bases, and
3. an application of the tool in our motivational setting of situation recognition for urban automated driving.

2   Related Work

We previously claimed that for our motivational domain of ADS development the usefulness of temporal logics (TLs) and related mechanisms – e.g., regular expressions – for scenario extraction has been recognized, which is supported by the literature [26, 31, 18, 16]. More specifically, work exists in specifying behavioral requirements, e.g., based on traffic rules, using TLs [1, 33, 19]. However, none of these approaches formally incorporate an ontology. In general, the importance of ontologies in automated driving is recognized, see, e.g., ASAM OpenXOntology [7] for an international standardization project as well as Westhofen et al. [42] and Zipfl et al. [44] for non-systematic reviews. Some ontological approaches are in fact based on DLs [27]. However, we are not aware of work within the automotive domain that uses DLs and TLs for analyzing temporal traffic data.

On the theoretical side, a plethora of temporal DLs have been introduced [2, 32, 5], also on finite traces [6]. These classical combinations were not conceived in a query answering context, so more recently, several frameworks for addressing that have been introduced [3]. We mention the most important ones here. There is work on ontologies formulated in the lightweight (i.e., comparatively inexpressive) DLs DL-Lite [12, 38] and $\mathcal{EL}$ [13, 22]. For expressive DLs, an important line of work theoretically examines answering temporal conjunctive queries – essentially infinite-time LTL over conjunctive queries – over temporal knowledge bases with the ontology language ranging from $\mathcal{ALC}$ [8] to $\mathcal{SHQ}$ [30, 9]. Related, but orthogonal to combinations of DLs with TLs, are combinations of Datalog with TLs. This line of research started around 1990 with Datalog1S [15], and lead to other combinations [14, 39] for which also tools exist [40].

3   Formal Foundations

We introduce the formal foundations of the relevant DLs and their temporal extension. For the sake of simplicity, we focus on the ontology language $\mathcal{ALC}$, which is a prototypical language in the class of expressive DLs. However, our approach generalizes to (and is actually implemented for) the more expressive logic $\mathcal{SRIQ}^{(\mathcal{D})}$, cf. Horrocks et al. for further reference on this DL fragment [24].

We start with an introduction to non-temporal knowledge bases which we later use as a foundation for defining the temporal case. As sketched in Section 1, in $\mathcal{ALC}$ we can describe the relationship of roles and concepts in an ontology $\mathcal{O}$ and assert individuals to these concepts and roles in a database $\mathcal{D}$. Any knowledge base is thus a tuple $(\mathcal{O}, \mathcal{D})$ and relies upon concept, role, and individual names. For the remainder, we fix countably infinite supplies N_C, N_R, N_I of concept, role, and individual names, respectively. An $\mathcal{ALC}$-concept description C is formed according to $C ::= A \mid \neg C \mid C \sqcap C \mid C \sqcup C \mid \forall r.C \mid \exists r.C$ where A ranges over N_C and r ranges over N_R. We can thus compose new concepts using negation, intersection, and union. For a role r, we moreover allow for universal (enforcing a concept to only have r-successors in C) and existential quantification (enforcing a concept to have an r-successor in C). Section 1 already introduced an example of an existentially quantified role using $\exists \texttt{drives.Vehicle}$ – the concept of all things driving some vehicle. An ontology is a set of concept inclusions $C \sqsubseteq D$ for $\mathcal{ALC}$-concepts C and D, denoting subsumption of the concept C to the concept D. We write $C \equiv D$ (concept equivalence) for $C \sqsubseteq D$ and $D \sqsubseteq C$. Again, the introduction used $\texttt{Human} \sqcap \exists \texttt{drives.Vehicle} \equiv \texttt{Driver}$ as an example for concept equivalence. The data is a set of facts of the form $A(a)$ and $r(a, b)$ for $a, b \in N_I, r \in N_R$, and $A \in N_C$, hence assigning individuals to concepts and roles. We denote the set of individuals that occur in $\mathcal{D}$ by $\mathsf{Ind}(\mathcal{D})$. The introductory example of Section 1 used the set of individuals $\{\texttt{h}, \texttt{v}\}$ and asserted the role $\texttt{drives}(\texttt{h}, \texttt{v})$.

The semantics of ontologies and data is defined via interpretations $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ of a domain $\Delta^{\mathcal{I}}$ and a mapping $\cdot^{\mathcal{I}}$ that assigns a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ to every $A \in N_C$, a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ to every role name $r \in N_R$, and an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ to every $a \in N_I$ [10, Chapter 2.2]. As to incorporate $\mathcal{ALC}$-concept descriptions, the interpretation function is inductively defined as:

$$(\neg C)^{\mathcal{I}} := \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$$
$$(C \sqcap D)^{\mathcal{I}} := C^{\mathcal{I}} \cap D^{\mathcal{I}}$$
$$(C \sqcup D)^{\mathcal{I}} := C^{\mathcal{I}} \cup D^{\mathcal{I}}$$
$$(\forall r.C)^{\mathcal{I}} := \{c \in \Delta^{\mathcal{I}} \mid \forall d \in \Delta^{\mathcal{I}}. (c, d) \in R^{\mathcal{I}} \rightarrow d \in C^{\mathcal{I}}\}$$
$$(\exists r.C)^{\mathcal{I}} := \{c \in \Delta^{\mathcal{I}} \mid \exists d \in \Delta^{\mathcal{I}}. (c, d) \in R^{\mathcal{I}} \wedge d \in C^{\mathcal{I}}\}$$

Then, we say $\mathcal{I} \models C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, $\mathcal{I} \models A(a)$ if $a^{\mathcal{I}} \in A^{\mathcal{I}}$, and $\mathcal{I} \models r(a, b)$ if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$. As to lift these definitions to ontologies and data, we write $\mathcal{I} \models \mathcal{O}$ and $\mathcal{I} \models \mathcal{D}$ if $\mathcal{O}$ resp. $\mathcal{D}$ satisfy *all* concept inclusions in $\mathcal{O}$ resp. assertions in $\mathcal{D}$. Finally, for a complete knowledge base, we define $\mathcal{I} \models (\mathcal{O}, \mathcal{D})$ if $\mathcal{I} \models \mathcal{O}$ and $\mathcal{I} \models \mathcal{D}$. More details on the semantics of DLs are given by Baader et al. [10].

We now extend this definition of non-temporal knowledge bases to the temporal case, where a knowledge base consists of an ontology $\mathcal{O}$ and a finite sequence of assertions that describe the databases over time.

Definition 1 (Temporal Knowledge Base). *A* temporal knowledge base (KB) *is a tuple* $\mathcal{K} = (\mathcal{O}, (\mathcal{D}_i)_{i \in \{0,\ldots,n\}})$ *where* $\mathcal{O}$ *is an ontology and each* $\mathcal{D}_i$ *is a database.*

Their semantics is defined by temporal interpretations using the non-temporal case as its basis.

Definition 2 (Temporal Interpretation). *A temporal interpretation* $\mathfrak{I}$ *is a finite sequence* $\mathfrak{I} = (\mathcal{I}_i)_{i \in \{0,\ldots,m\}}$ *of interpretations over a fixed domain* Δ *such that* $a^{\mathcal{I}_i} = a^{\mathcal{I}_j}$, *for all* $a \in \mathsf{N}_\mathsf{I}$ *and* $0 \le i, j \le m$. *We call* $\mathfrak{I}$ *a* model *of the temporal KB* $(\mathcal{O}, (\mathcal{D}_i)_{i \in \{0,\ldots,n\}})$, *written* $\mathfrak{I} \models \mathcal{K}$, *if* $m = n$ *and* $\mathcal{I}_i \models \mathcal{D}_i$ *and* $\mathcal{I}_i \models \mathcal{O}$, *for all* $i \in \{0,\ldots,n\}$.

The assumption that all interpretations share a common domain is called *constant domain assumption*. We define next the language MTCQ that we use to query temporal KBs. It is a combination of standard conjunctive queries with temporal operators inspired by MLTL [29].

Definition 3 (Syntax of MTCQs). *Let* N_V *be a countably infinite set of variable names. A* conjunctive query (CQ) φ *is an expression of the form* $\varphi(\vec{x}) = \exists \vec{y}.\psi(\vec{x}, \vec{y})$ *where* $\vec{x}, \vec{y}$ *are tuples of variables from* N_V *and* ψ *is a conjunction of concept atoms* $A(t)$ *and role atoms* $r(t,t')$ *with* $A \in \mathsf{N}_\mathsf{C}$, $r \in \mathsf{N}_\mathsf{R}$, *and* $t, t' \in \vec{x} \cup \vec{y} \cup \mathsf{N}_\mathsf{I}$. *Metric temporal conjunctive queries (MTCQs)* Φ *are built from CQs using* negation $\neg\Phi$, *conjunction* $\Phi \wedge \Phi'$, *and two versions of* until, $\Phi\mathcal{U}\Phi'$ *and* $\Phi\mathcal{U}_{[a,b]}\Phi'$ *for* $a, b \in \mathbb{N}$. *We denote with* $\mathsf{Ind}(\Phi)$ *the set of individuals and* $\mathsf{Var}(\Phi)$ *the set of variables in an MTCQ* Φ.

Note that we extend MLTL by borrowing the unconstrained until operator from LTL, because it is a frequent operator in practice. Additionally, it allows for a more direct translation to finite automata in our system presented later on. We call the variables $\vec{x}$ the *answer variables* and $\vec{y}$ the *quantified variables*. An MTCQ is *Boolean* if it does not have answer variables. The semantics of Boolean CQs is defined in terms of matches into interpretations.

Definition 4 (Semantics of Boolean CQs). *For a Boolean conjunctive query* φ *and an interpretation* $\mathcal{I}$, $\mathcal{I} \models \varphi$ *iff there exists a function* $\pi\colon \mathsf{Var}(\varphi) \cup \mathsf{Ind}(\varphi) \to \Delta^{\mathcal{I}}$ *with 1.* $\pi(a) = a^{\mathcal{I}}$ *for all* $a \in \mathsf{Ind}(\varphi)$, *2.* $\pi(t) \in \mathsf{C}^{\mathcal{I}}$ *for all* $\mathsf{C}(t)$ *in* φ, *and 3.* $(\pi(t), \pi(t')) \in \mathsf{r}^{\mathcal{I}}$ *for all* $\mathsf{r}(t,t')$ *in* φ.

Hence, an interpretation satisfies a Boolean CQ if the interpretation can respect its constraints. Boolean CQs form the basis for the semantics of Boolean MTCQs.

Definition 5 (Semantics of Boolean MTCQs). *Let* $\mathfrak{I} = (\mathcal{I}_i)_{i \in \{0,\ldots,m\}}$ *be a temporal interpretation and* $i \in \{0,\ldots,m\}$. *The semantics of Boolean MTCQs is given by structural induction:*

- $\mathfrak{I}, i \models \Phi$ *iff* $\mathcal{I}_i \models \Phi$, *if* Φ *is a Boolean CQ;*

- $\mathfrak{I}, i \models \neg\Phi$ iff $\mathfrak{I}, i \not\models \Phi$;
- $\mathfrak{I}, i \models \Phi_1 \wedge \Phi_2$ iff $\mathfrak{I}, i \models \Phi_1$ and $\mathfrak{I}, i \models \Phi_2$;
- $\mathfrak{I}, i \models \Phi_1 \mathcal{U}_{[a,b]} \Phi_2$ iff there is a $k \in [a,b]$ with $i+k \le m$ such that $\mathfrak{I}, i+k \models \Phi_2$ and $\mathfrak{I}, i+j \models \Phi_1$, for all $j \in [a,k)$;
- $\mathfrak{I}, i \models \Phi_1 \mathcal{U} \Phi_2$ iff there is a $k \in [i,m]$ such that $\mathfrak{I}, k \models \Phi_2$ and $\mathfrak{I}, j \models \Phi_1$, for all $j \in [i, i+k)$.

We allow the typical abbreviations $\Phi \vee \Phi'$ for $\neg(\neg\Phi \wedge \neg\Phi')$, false for $\exists x. A(x) \wedge \neg\exists x. A(x)$ for some $A \in \mathsf{N_C}$, true for $\neg$false, $\Diamond_{[a,b]}\Phi$ for true$\mathcal{U}_{[a,b]}\Phi$, $\Diamond\Phi$ for true$\mathcal{U}\Phi$, $\Box_{[a,b]}\Phi$ for $\neg\Diamond_{[a,b]}\neg\Phi$, and $\Box\Phi$ for $\neg\Diamond\neg\Phi$. The *strong next-operator* is defined as $\bigcirc\Phi \equiv \Diamond_{[1,1]}\Phi$ and *weak next* as $\bullet\Phi \equiv \Box_{[1,1]}\Phi$. Note that finite trace semantics exhibit some non-obvious behaviors, e.g., $\Diamond\Box\Phi$ is equivalent to $\Box\Diamond\Phi$ [17].

A central problem over Boolean MTCQs is *entailment*: For a temporal KB $\mathcal{K}$ and an MTCQ Φ, we say $\mathcal{K} \models \Phi$ if for all temporal interpretations $\mathfrak{I}$ with $\mathfrak{I} \models \mathcal{K}$ also $\mathfrak{I}, 0 \models \Phi$ holds. For example, for $\mathcal{K}^{ex}$ from Section 1, it holds that $\mathcal{K}^{ex} \models \Diamond\neg\texttt{Pedestrian(h)}$ as for any temporal interpretation $\mathfrak{I} = (\mathcal{I}_0, \mathcal{I}_1)$ with $\mathcal{I}_0 \models \mathcal{O}^{ex}$ and $\mathcal{I}_0 \models \mathcal{D}_0^{ex}$, it must also hold that $\texttt{h}^{\mathcal{I}_0} \in (\neg\texttt{Pedestrian})^{\mathcal{I}_0}$ due to the fact that $\texttt{h}$ is inferred to be a driver and thus cannot be a pedestrian.

We remark that this semantics is closely related to the one over temporal conjunctive queries (TCQs) introduced by Baader et al. [8] to query temporal KBs over arbitrary models, i.e., not restricted to mission time. In fact, it is not difficult to see that entailment $\mathcal{K} \models \Phi$ for Boolean MTCQs Φ can be reduced to deciding whether $\mathcal{K}$ entails $\widehat{\Phi}$ in the sense of Baader et al. [8] for some TCQ $\widehat{\Phi}$ that can be computed in polynomial time from Φ; we denote the latter entailment relation with $\mathcal{K} \models^{\mathrm{BBL}} \widehat{\Phi}$. In the mentioned paper it is also shown that the latter entailment problem is in ExpTime. Together with the ExpTime-lower bound for subsumption in $\mathcal{ALC}$ this shows that MTCQ entailment is ExpTime-complete. Of course, the complexity is potentially higher for ontology languages beyond $\mathcal{ALC}$. Finally, if in place of CQs in MTCQs we allow for $\mathcal{ALC}$-concepts, the resulting language can be embedded into the metric temporal DLs discussed by Gutiérrez-Basulto et al. [23].

While Boolean MTCQ entailment is the natural problem to consider for complexity analysis, a practical system needs support for *answering non-Boolean MTCQs*, which is defined based on entailment. Let $\mathcal{K} = (\mathcal{O}, (\mathcal{D}_i)_{i \in \{0,\dots,n\}})$ be a temporal KB, $\Phi(\vec{x})$ an MTCQ with answer variables $\vec{x}$, and $\vec{a}$ a tuple of individuals from $\mathcal{K}$, i.e., $\vec{a} \subseteq \mathsf{Ind}(\mathcal{K}) :- \bigcup_{i=0,\dots,n} \mathsf{Ind}(\mathcal{D}_i)$. We call $\vec{a}$ a *certain answer to $\Phi(\vec{x})$ over $\mathcal{K}$* if $\mathcal{K} \models \Phi(\vec{a})$. Here, $\Phi(\vec{a})$ is the uniform replacement of the variables in $\vec{x}$ by the individual names in $\vec{a}$, leading to a Boolean MTCQ. Our main reasoning task is to compute the set $\mathsf{cert}_{\mathcal{K}}(\Phi)$ of certain answers of Φ over $\mathcal{K}$. Section 1 gives an example for this set: $\mathsf{cert}_{\mathcal{K}^{ex}}(\Diamond\neg\texttt{Pedestrian}(x)) = \{\texttt{h}\}$.

4   Computing Certain Answers in Practice

We start with noting that to compute $\mathsf{cert}_{\mathcal{K}}(\Phi)$, it is not sufficient to answer all of Φ's CQs at time i and combine them inductively according to the semantics

due to the presence of disjunction in our query language. An example is the MTCQ $\Phi_\vee(x) := \mathtt{B}(x) \vee \mathtt{C}(x)$ over the temporal KB $\mathcal{K}_\vee := (\mathtt{A} \sqsubseteq \mathtt{B} \sqcup \mathtt{C}, (\mathtt{A}(\mathtt{a})))$, where $\mathsf{cert}_{\mathcal{K}_\vee}(\Phi_\vee) = \{(\mathtt{a})\}$. A separate check of $\mathtt{B}(x)$ and $\mathtt{C}(x)$ returns no answer, and inductive combination falsely yields no answer as well. This issue explains the restriction to conjunctions in existing CQ answering implementations over expressive DLs, as complexity is reduced and various optimizations can be employed. Therefore, and in contrast to both $\mathrm{LTL_f}$ over propositional atoms and CQ answering, we require a more involved procedure for checking MTCQs.

The correct but naïve way to compute $\mathsf{cert}_{\mathcal{K}}(\Phi)$ is to enumerate all candidate answers $\vec{a} \subseteq \mathsf{Ind}(\mathcal{K})$ and decide whether $\mathcal{K} \models^{\mathrm{BBL}} \widehat{\Phi}(\vec{a})$ via the algorithms provided by Baader et al. [8] (for the temporal aspects) and Horrocks and Tessaris [25] (for answering disjunctions of conjunctive queries). This, however, suffers from several problems. First, there are potentially many answer candidates since the number of relevant tuples is exponential in the arity of the query Φ. Second, while the mentioned algorithm for deciding $\models^{\mathrm{BBL}}$ is useful for a complexity analysis, it does not lend itself to a direct implementation. Finally, the algorithm of Baader et al. works over unrestricted models and is thus more difficult to implement. This section provides the foundations for the algorithm that we implemented in our tool and the central improvements needed to make it work in practice.

As MTCQs are closed under negation, entailment is just the complement of *satisfiability*: a Boolean MTCQ Φ is satisfiable w.r.t. a temporal KB $\mathcal{K}$ if there is a model $\mathfrak{I}$ of $\mathcal{K}$ with $\mathfrak{I}, 0 \models \Phi$. As $\mathcal{K} \models \Phi$ iff $\neg\Phi$ is unsatisfiable w.r.t. $\mathcal{K}$, we can, for the sake of convenience, focus on satisfiability in the following.

We need some preliminary notions. Given an MTCQ Φ (possibly with answer variables), we denote with $\mathsf{CQ}(\Phi)$ the set of all CQs in Φ. The *propositional abstraction* $\mathsf{PA}(\Phi)$ of Φ is the replacement of each $\varphi \in \mathsf{CQ}(\Phi)$ with a propositional variable p_φ. Note that the propositional abstraction of an MTCQ is an MLTL formula potentially with an unconstrained until, which is the underlying temporal formalism. This TL is interpreted over finite words $P_0 \cdots P_n$ where each P_i specifies the propositional variables that are satisfied at time point i. Boolean operators are interpreted as usual and temporal operators $\mathcal{U}$ and $\mathcal{U}_{[a,b]}$ are interpreted in line with Definition 5. The following characterization of satisfiability is easy to prove from the definitions.

Lemma 1. *For a Boolean MTCQ Φ and a temporal KB $\mathcal{K} = (\mathcal{O}, (\mathcal{D}_i)_{i \in \{0,\ldots,n\}})$, Φ is satisfiable w.r.t. $\mathcal{K}$ iff there is a sequence $X_0, \ldots, X_n$ of subsets of $\mathsf{CQ}(\Phi)$ such that:*

1. *there are interpretations $\mathcal{I}_0, \ldots, \mathcal{I}_n$ over the same domain such that, for all $i \in \{0, \ldots, n\}$, we have $\mathcal{I}_i \models \mathcal{O}$, $\mathcal{I}_i \models \mathcal{D}_i$, and $\mathcal{I}_i \models \varphi$ for every $\varphi \in X_i$, and $\mathcal{I}_i \not\models \varphi$ for every $\varphi \in \mathsf{CQ}(\Phi) \setminus X_i$, and*
2. *$(\{p_\varphi \mid \varphi \in X_i\})_{i \in \{0,\ldots,n\}}$ satisfies $\mathsf{PA}(\Phi)$.*

Intuitively, Lemma 1 splits the problem of deciding MTCQ satisfiability into separate DL and TL tasks which are only connected by the sets of CQs $X_0, \ldots, X_n$.

Lemma 1 can be further refined as follows. The requirement that all interpretations $\mathcal{I}_0, \ldots, \mathcal{I}_n$ be over the same domain can be dropped without compromising correctness. Indeed, we can combine $\mathcal{I}_0, \ldots, \mathcal{I}_n$ witnessing Point 1 in Lemma 1 but with potentially different domains into $\mathcal{I}'_0, \ldots, \mathcal{I}'_n$ with the same domain using a standard argument, cf. the proof of Theorem 5.21 by Lippmann [30]: Since $\mathcal{ALC}$ cannot enforce finite models, we can assume that each $\mathcal{I}_i$ is infinite. By the downward Löwenheim-Skolem-Theorem, we can assume that the $\mathcal{I}_i$ are countably infinite and thus have the same domain. It remains to identify the interpretation of the individual names. Note that the argument goes through for more expressive logics such as $\mathcal{SRIQ}^{(\mathcal{D})}$.

Lemma 2. *Lemma 1 remains valid when "over the same domain" is dropped from Point 1.*

Hence, the checks at each time in (the modified) Point 1 are independent. It remains to show how we can implement the check of Point 1, which includes negated CQs. By the natural connection between satisfiability and entailment, we can leverage an engine for answering disjunctions of CQs over non-temporal $\mathcal{ALC}$ KBs for this, i.e., computing $\mathsf{cert}_{\mathcal{K}}(\varPhi)$ for $\mathcal{K} = (\mathcal{O}, \mathcal{D})$ and $\varPhi$ a disjunction of CQs. For doing so, we associate with every Boolean CQ φ its *canonical database* $\mathcal{D}_\varphi$ which is just the set of all conjuncts that occur in φ. (For the sake of simplicity, we allow variable names from φ as individual names in $\mathcal{D}_\varphi$.) We then exploit the following observation.

Observation 1 *Let X be a set of Boolean CQs, let $\mathcal{O}$ be an $\mathcal{ALC}$-ontology and $\mathcal{D}$ the data. Then the following are equivalent for every subset $Z \subseteq X$:*

(a) *There is a model $\mathcal{I}$ of $\mathcal{O}$ and $\mathcal{D}$ such that $\mathcal{I} \models \varphi$ for every $\varphi \in Z$, and $\mathcal{I} \not\models \varphi$ for every $\varphi \in X \setminus Z$.*
(b) *$(\mathcal{O}, \mathcal{D}') \not\models \bigvee_{\varphi \in X \setminus Z} \varphi$ where $\mathcal{D}'$ is the union of $\mathcal{D}$ with $\mathcal{D}_\varphi$ for each $\varphi \in Z$ (with variables across different $\mathcal{D}_\varphi$ suitably renamed).*

Thus, to check the modified Point 1 for some time point (a condition of shape (a) in the above Lemma), we can check its reformulation as (b) using a (non-temporal) query engine for disjunctions of CQs. As demonstrated by the exemplary query $\varPhi_\vee(x)$, this is, however, more involved than answering each disjunction separately, a problem already known to the DL community. For correctly answering such disjunctions of CQs, we require a reformulation in of the disjunction into conjunctive normal form, and then answer each conjunct separately as described by Horrocks et al. [25]. For $P \subseteq \{p_\varphi \mid \varphi \in \mathsf{CQ}(\varPhi)\}$, we define $\mathsf{VAL}_\varPhi^i(P)$ as true iff. $\mathcal{O}, \mathcal{D} := \mathcal{D}_i, Z := \{\varphi \mid p_\varphi \in P\}, X := \mathsf{CQ}(\varPhi)$ pass the test in Point (b), and thus the modified Point 1.

To implement Point 2, we exploit that for each MLTL formula χ over some set of propositions Σ, one can compute an equivalent LTL_f (LTL over finite traces) formula χ' over Σ [29] which in turn can be transformed into a finite automaton (FA) $\mathfrak{A}_\chi$ over 2^Σ which recognizes precisely the models of χ' and thus of χ [17]. Both these transformations are not polynomial and there is, in general, no efficient conversion of an MLTL formula to an FA. However, since
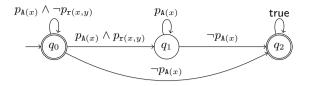
Fig. 1. FA for $\mathsf{PA}(\neg(\Box\mathtt{A}(x) \wedge \Diamond\mathtt{r}(x,y)))$.

Algorithm 1 Computing certain answers to MTCQs.

Input: MTCQ $\Phi(\vec{x})$, temporal KB $\mathcal{K} = (\mathcal{O}, (\mathcal{D})_{i \in \{0,\dots,n\}})$
Output: $\mathsf{cert}_{\mathcal{K}}(\Phi)$.

 1: $\mathfrak{D} := \textsc{Construct\_FA}(\mathsf{PA}(\neg\Phi))$;
 2: // states Q, initial state q_0, final states F, transitions Δ
 3: $C := \mathsf{ind}(\mathcal{D})^k$ where $k = |\vec{x}|$
 4: Initialize $S(\vec{a}, 0) := \{q_0\}$ for all $\vec{a} \in C$
 5: **for** $i := 1$ to $n + 1$ **do**
 6:    **for** $\vec{a} \in C$ **do**
 7:        $S(\vec{a}, i) := \emptyset$
 8:        **for** $q \in S(\vec{a}, i - 1)$ **do**
 9:            $S(\vec{a}, i) := S(\vec{a}, i) \cup \{q' \mid (q, X, q') \in \Delta, \mathsf{VAL}^{i-1}_{\neg\Phi(\vec{a})}(X)\}$
10:        **end for**
11:    **end for**
12: **end for**
13: **return** $\{\vec{a} \in C \mid S(\vec{a}, n + 1) \cap F = \emptyset\}$;

---

queries are often small in practice, this is still feasible. For example, the minimal FA for $p_1 \mathcal{U}_{\leq a} \Diamond_{\leq b} p_2$ has $a + b + 3$ states. Figure 1 shows the FA for answering the simple MTCQ $\Phi_{ex}(x, y) = \Box\mathtt{A}(x) \wedge \Diamond\mathtt{r}(x, y)$. Note that the transitions are labeled with Boolean formulas over the propositions indicating a transition for each model of the formula, which can be exponentially more succinct.

What was said so far suggests the basic procedure for computing $\mathsf{cert}_{\mathcal{K}}(\Phi)$ that is depicted in Algorithm 1. It considers for each answer candidate $\vec{a}$ all possible 'runs' $X_0, \dots, X_n$ in a step-by-step fashion and checks (modified) Points 1 and 2 after each step; the set $S(\vec{a}, i)$ contains all states the FA corresponding to $\neg\Phi(\vec{a})$ can reach after i steps. The central test happens in Line 7 and is given here for the direct encoding of the transitions; it can easily be adapted for the mentioned succinct encoding. The algorithm returns all $\vec{a}$ for which no final state is reachable after $n + 1$ steps. Applied to the example FA in Figure 1 and a candidate answer (a, b) this means that the FA ends up in state q_1 in all possible runs, according to the temporal KB. The only way to achieve this is for the FA to not stay in q_0 or q_2. For this, it has to eventually change from q_0 to q_1 by having neither $\mathtt{A}(a) \wedge \neg\mathtt{r}(a, b)$ nor $\neg\mathtt{A}(a)$ but $\mathtt{A}(a) \wedge \mathtt{r}(a, b)$ satisfiable. The FA shall then stay solely in q_1 with only $\mathtt{A}(a)$ satisfiable for the remainder. Clearly, in this case (a, b) is a certain answer.

4.1   Improvements

Some standard improvements over Algorithm 1 are applicable, e.g., to work directly on a minimal FA. However, this does not yet address the problem of the many answer candidates to consider, of which, in practice, only few will be entailed. Algorithm 1 considers each candidate individually, which is inefficient since similar tasks are repeatedly executed. We instead leverage existing systems that implement efficient algorithms specifically tailored towards answering CQs over standard (non-temporal) KBs. As an example, consider again the FA in Figure 1. Observe that $q_2 \in S(\vec{a}, i)$ for all $\vec{a}$, i for which $(\mathcal{O}, \mathcal{D}_{i-1}) \not\models \mathtt{A}(\vec{a})$. Indeed, $\neg \mathtt{A}(\vec{a})$ is satisfiable w.r.t. $(\mathcal{O}, \mathcal{D}_{i-1})$, for those $\vec{a}, i$. Since q_2 is a sink, this allows us to instantly reject all non-answers to $\mathtt{A}(x)$. We now generalize this to extract certain (non-)answers by answering the CQs occurring in the edges.

The main idea is to perform an under-approximating traversal of the FA prior to Algorithm 1. More concretely, we use CQ answering to construct sets $R(\vec{a}, i) \subseteq S(\vec{a}, i)$ and $U(\vec{a}, i) \subseteq Q \setminus S(\vec{a}, i)$ that under-approximate the reachable and unreachable states, respectively, for a candidate $\vec{a}$ at time i. This serves two purposes. First, we can already extract some certain answers from U and some certain non-answers from R, namely the sets $\{\vec{a} \in C \mid U(\vec{a}, n+1) \supseteq F\}$ and $\{\vec{a} \in C \mid R(\vec{a}, n+1) \subseteq F\}$, respectively. These candidates are not considered anymore during the run of Algorithm 1. Second, we are able to re-use cached answers to CQs in the first traversal during Algorithm 1.

We now describe how to construct the sets R and U during FA traversal. $R(\vec{a}, 0)$ is initialized as $\{q_0\}$ and $U(\vec{a}, 0)$ is initialized as $Q \setminus \{q_0\}$, for all $\vec{a}$. For the update step with $i > 0$, we assume for all states q_k, q_l to have succinctly encoded edges $\alpha_{k,l} := \bigwedge_{p_\varphi \in P_0} \neg p_\varphi \wedge \bigwedge_{p_\varphi \in P_1} p_\varphi$ for some sets $P_0, P_1 \subseteq P$, as already used in Figure 1. When examining such an edge in the FA at time i, we use a CQ engine on $\mathcal{K}_i := (\mathcal{O}, \mathcal{D}_i)$ to compute $\mathsf{cert}_{\mathcal{K}_i}(\varphi)$ for all $\varphi \in \{\psi \mid p_\psi \in P_0\} \cup \{\bigwedge_{p_\psi \in P_1} \psi\}$. From these sets, we are able to extract information on the relevant queries:

1. for all $\vec{a} \notin \mathsf{cert}_{\mathcal{K}_i}(\varphi)$: $\neg\varphi(\vec{a})$ is *satisfiable* w.r.t. $\mathcal{K}_i$;
2. for all $\vec{a} \in \mathsf{cert}_{\mathcal{K}_i}(\varphi)$: $\varphi(\vec{a})$ is *satisfiable* and $\neg\varphi(\vec{a})$ is *unsatisfiable* w.r.t. $\mathcal{K}_i$.

We transfer this knowledge about the (un-)satisfiability of $\varphi(\vec{a})$ and $\neg\varphi(\vec{a})$ to the edges $\alpha_{k,l}$. *Satisfiability* knowledge is transferable if $q_k \in R(\vec{a}, i-1)$ and $\alpha_{k,l} = p_\varphi$ resp. $\alpha_{k,l} = \neg p_\varphi$. We then add q_l to $R(\vec{a}, i)$. *Unsatisfiability* knowledge on $\neg\varphi(\vec{a})$ is transferable if $\alpha_{k,l}$ contains $\neg p_\varphi$. Adding unsatisfiability knowledge to U requires adaptations. Firstly, we can only add q_l to $U(\vec{a}, i)$ if *all* other edges $\alpha_{j,l}$ to q_l also agree on unsatisfiability of $\vec{a}$ at time i, i.e., they contain some $\neg p_{\varphi'}$ for which $\varphi'(\vec{a})$ is known to be unsatisfiable or $q_j \in U(\vec{a}, i-1)$. Secondly, unsatisfiability generates new satisfiability information: for a state q_k with successors $q_{l_1}, \ldots, q_{l_h}$ we know that $\{q_{l_1}, \ldots, q_{l_{h-1}}\} \subseteq U(\vec{a}, i)$ implies $q_{l_h} \in R(\vec{a}, i)$. Together with the described acceptance condition, the sets $R(\vec{a}, n+1)$ and $U(\vec{a}, n+1)$ deliver an under-approximation of the certain (non-)answers.

4.2 Our System

We implemented this approach as a module in the DL reasoner OPENLLET [37]. The implementation is available at `https://github.com/lu-w/topllet`. Our module does not support full MTCQs yet. Instead of allowing arbitrary CQs as atoms, we allow the subclass tCQ of CQ which consists of all CQs φ s.t. in the graph $G_\varphi = (V, E)$ with $V = \mathsf{Var}(\varphi) \cup \mathsf{Ind}(\varphi)$ and $E = \{(t, r, t') \mid r(t, t') \in \varphi\}$ each vertex has at most one incoming edge and, if interpreted undirectedly, G is acyclic, i.e., the query graph is *tree-shaped*[4].

As a first necessary step, we denote with tMTCQ the subclass of MTCQ where each CQ is in fact a tCQ. The reasons for considering this query class are two-fold. First, most queries that occur in practice are tMTCQs. Second, tCQ answering can be implemented by a straightforward procedure of 'rolling-up' the query graph [25]. Therefore, OPENLLET already provides an tCQ-answering engine over $\mathcal{SROIQ}^{(\mathcal{D})}$ KBs, implementing many optimizations [36]. Moreover, the procedure can be adapted to answering disjunctions of tCQs as described by Horrocks and Tessaris [25], which required for our algorithm, cf. Point (b) in Observation 1.

As a first necessary step, we thus extended OPENLLET to being able to answer disjunctions of tCQs. For the construction of the FA, we implemented the conversion of MLTL to LTL_f described by Li et al. [29]: essentially, the intervals in $\mathcal{U}_{[a,b]}$ are encoded using sequences of the next-operator $\bigcirc$ of length a and b, respectively. We then rely on LYDIA, which converts LTL_f formulas to equivalent deterministic FA [20]. We extend and use the AUTOMATALIB [28] to access the resulting FA. We provide a test suite for our system to highlight correctness of the implemented algorithms.

5 Benchmarks

Our CQ answering approach motivates the need for empirical evaluation, for which ideally controlled real-world data is used. In fact, for one experiment, we obtained drone data from an intersection in Germany. These data turned out to be insufficient for a thorough evaluation, as they are proprietary and not scalable. This calls for synthetic yet realistic benchmark data that can be randomized, scaled in size, and are freely available for replicability. However, we are currently not aware of *any* public benchmark data on querying temporal KBs. The same was noted by the developers of METEOR, where data of the Lehigh University Benchmark [21] are extended with random intervals to enable an evaluation on the OWL RL fragment of LUBM. Unfortunately, a random extension of a non-temporal benchmark might not reflect actual temporal data, e.g., in continuity of concepts over time, and thus might not transfer to real-world applications. As our final contribution, we hence present the Traffic Ontology Benchmark (TOBM), a benchmark generator for scenarios of automated driving applications that mimics

---

[4] This constraint allows us to perform the rolling-up procedure on the BCQs of the FA. However, it is actually just a sufficient condition for rolling-up. More precisely, we require the FA to contain only BCQs where each negated query is a tCQ.
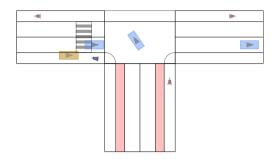
Fig. 2. A scene of the T-crossing scenario sampled from TOBM.

real-world data and enables to evaluate tools on temporal KBs, including MTCQ answering. The tool is available at `https://github.com/lu-w/tobm`.

For the ontology we rely on the publicly available Automotive Urban Traffic Ontology (A.U.T.O.) [42, Section 5]. It is a conglomerate of $\mathcal{SRIQ}^{(\mathcal{D})}$ ontologies for the traffic domain and related fields, and currently consists of 1449 axioms over 676 concepts and 213 roles. A.U.T.O. was already successfully used for analyzing real-world traffic data from drone recordings [42, Section 8].

The benchmark generator creates temporal data for A.U.T.O. with individuals scaling linear to some $N > 0$. A seed S can be used for pseudo-randomization. From both parameters, it generates scenarios of a certain length (by default, 20 seconds). These can be sampled from two settings:

1. A T-crossing setup with parking vehicles, a pedestrian crossing, bikeway lanes, pedestrians, bicyclists, and passenger cars (cf. Figure 2). It has $8 \cdot N + 22$ individuals.
2. An X-crossing of two urban roads with traffic signs and dysfunctional traffic lights. Compared to the T-crossing, there are no bicyclists and $5 \cdot N + 69$ individuals.

The scenarios are created based on behavior models for pedestrians, bicyclists, and passenger cars. Passenger cars and bicyclists drive up to a speed limit if their front area is free, otherwise they use a following mode. Vehicles yield on a predicted intersecting path. Moreover, a random successor lane is selected when turning at intersections, giving a turning signal with a probability of 3% each time point. Pedestrians follow their walkway, but can randomly initiate road crossing with a probability of 0.7%. We give a visualization of two exemplary scenarios can be found in the linked repository.

Our implementation models temporal KBs as a list of OWL2-files for the data, each importing a shared ontology. Geometrical data are abstracted to spatial predicates (e.g., `is_in_front_of`) in a pre-processing step. For $S = 0$, $N = 3$, and 20 seconds sampled with 10 Hertz on the T-crossing setting, this results in a data sequence with 46 individuals and $647\,847$ assertions in total (approx. $3\,239$ per time point) with constant assertions only counted once.

6   Evaluation

We now examine practical feasibility of our system by an evaluation on TOBM, answering the following questions:

1. Is the approach applicable to practical, a-posteriori situation recognition tasks (such as evaluating test data) with larger numbers of assertions?
2. What is the impact of our improvement of leveraging CQ answering on overall applicability?
3. In practical settings, how much satisfiability knowledge can be generated by CQ answering?

As inputs, we sampled TOBM with $S = 0$ and $N \in \{1, \ldots, 5\}$ for both the X- and T-crossing. We fix a 20 second duration with ten Hertz, as our algorithm performs linear in N. The supplementary artifact provides both the benchmarks and a wrapper around TOBM for reproducible re-generation. We used four queries (given in the supplementary artifact) asking for: intersecting paths with VRUs (Φ_1), passing of parking vehicles on two-lane roads (Φ_2), vehicles turning right (Φ_3), and vehicles changing lanes without signals (Φ_4), where Φ_1, Φ_2, and Φ_3 have two and Φ_4 has three answer variables. The corresponding FAs have 8 (Φ_1), 4 (Φ_2, Φ_4), and 3 (Φ_3) states. Our tool is executed once per benchmark and query combination, as deviations are not be expected due to determinism, on an Intel Core i9-13900K with 64 GB RAM and a time limit of ten hours per run, using a Windows Subsystem for Linux 1 on a Windows 10 host. The input files and tool, with the exact version and configuration used for benchmarking, are available online [41].
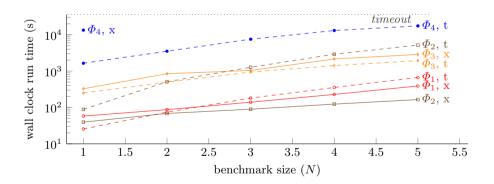

Fig. 3. Wall clock running times of benchmark queries Φ_i, $i \in \{1, \ldots, 4\}$ and the T- (t) resp. X-crossing (x) of size N.

For the first question, we show wall clock running times of our improved algorithm in Figure 3. We exclude parsing and loading of queries and KBs as we aim to only evaluate our algorithm. Running times indicate an exponential dependency on the data size. There are also dependencies on the benchmark

type, e.g., for Φ_2, where the non-existence of parking vehicles on the X-crossing improves performance, and Φ_4, where more lanes on the X-crossing increases running time. This answers the first question positively, as our approach terminates in minutes to hours, with the lowest being 25.54 seconds for Φ_1 on the 20 second T-crossing scenario. However, the timeout was reached for Φ_4 on the X-crossing and $N \geq 2$ for reasons to be discussed later.
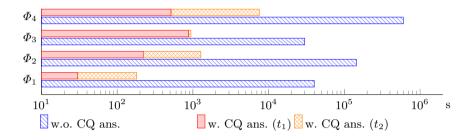

Fig. 4. Log-scaled running times with and without the CQ answering optimization enabled for the TOBM T-crossing $S = 0, N = 3$. Running times without the optimization are extrapolated after one hour.

The second question is addressed by comparing the running time of the improved algorithm to the basic algorithm from Algorithm 1. The results in Figure 4 show that the naïve approach fails for real-world data, even for two answer variables. Moreover, most of the time is still spent using the expensive, full semantics check despite iterating only through a fraction of all candidates (cf. Table 1). Hence, leveraging the CQ engine makes MTCQ answering practically feasible. However, some queries may trigger special cases in the optimizations of the CQ engine, leading to higher running times, e.g., role inclusion axioms for Φ_3.

The strong effect of leveraging CQ answering motivates deeper examination. For this final question, we show wall clock times of both the CQ answering run t_1 ('first run') and the full-semantics run t_2 ('second run') in Figure 4. The effect of CQ answering can be twofold: Firstly, a set of candidates can be excluded globally. Secondly, even if a candidate was not globally excluded, it generates 'local' (non-)answers that can be cached for subsequent checks of Point 1 of Lemma 1. We thus report both exclusions, averaged over all time points and checked edges at each time point, in Table 1. Moreover, one can ask whether the second run is actually worthwhile. Table 1 reports how many certain answers ($\mathsf{cert}_{\mathcal{K}}$) were already found in the first run ($\mathsf{cert}_{\mathcal{K}}^1$).

Our results show CQ answering to aid mainly by excluding candidates globally in a highly-optimized fashion, as it can resort to techniques like binary instance retrieval, and often avoids consistency checks [36]. Local exclusion has minor but non-negligible effects, e.g., avoiding on average 42 additional candidates for Φ_3. Moreover, all certain answers were already found in the first run, indicating suitability of using only the incomplete first run.

Table 1. Effects of CQ answering on MTCQ answering for the TOBM T-crossing $S = 0, N = 3$.

Query	Φ_1	Φ_2	Φ_3	Φ_4
Globally excluded candidates (%)	97.88	99.29	97.88	99.71
Globally and locally excluded candidates (%)	98.73	99.55	99.54	99.80
$|\mathsf{cert}_\mathcal{K}^1|/|\mathsf{cert}_\mathcal{K}|$	1	1	1	1

However, leveraging CQ answering has its limitations. For Φ_4 on the X-crossing and $N = 2$, the first run excluded 99.83% of all candidates after 2.38 minutes, leaving 960 candidates for the second run. However, this is no small task: for 200 time points in the data this leaves 180 seconds per time point to finish within 10 hours. Hence, each candidate must not take up more than 0.1875 seconds per time point on average, which entails checking multiple edges in multiple states. Experiments indicate each edge check to take a two-digit millisecond duration. Thus, to efficiently handle large candidate sets in the second run, we require further optimizations.

7   Conclusion

In this work, we introduced MTCQs as a suitable tool for situation recognition when testing requirements in complex operational domains, as illustrated by urban automated driving. Our tool, based on OPENLLET, brings MTCQ answering into practice by leveraging efficient CQ answering algorithms. Our custom benchmarks on safety-critical traffic situations show feasibility of our implementation for test evaluation settings and a potential to use our tool in other domains. These include risk assessments of other automated transportation systems, e.g., trams, maritime vessels, or delivery robots, and big-data analyses, e.g., process mining in business applications over intricate real-world structures.

As future work, we plan to investigate both practical optimizations and theoretical adaptations for increasing performance. For the former, it is interesting to *(i)* study how one can reuse query answers in consecutive time points given that potentially only small portions of the data change, *(ii)* identify fragments of MTCQs that can be answered more efficiently in practice (e.g., for runtime verification), and *(iii)* treat the spatial information more efficiently. On the theoretical side, it is interesting to study *rewriting* approaches, where the idea is to reduce the computation of certain answers to query evaluation in a target logic such as first-order logic (possibly with $+, <$) or DatalogMTL [39]. The benefit of such rewriting approaches is that one can leverage existing systems for evaluation in the target language. First-order rewritings have been studied in the context of more lightweight ontology and query languages [4]. While query rewritings need not exist in general (for complexity reasons), they might be very fruitful for practically occurring queries and ontologies.

References

1. Arechiga, N.: Specifying safety of autonomous vehicles in signal temporal logic. In: 2019 IEEE Intelligent Vehicles Symposium. pp. 58–63. IEEE, New York, USA (2019)
2. Artale, A., Franconi, E.: A survey of temporal extensions of description logics. Annals of Mathematics and Artificial Intelligence **30**(1), 171–210 (Jun 2000)
3. Artale, A., Kontchakov, R., Kovtunova, A., Ryzhikov, V., Wolter, F., Zakharyaschev, M.: Ontology-mediated query answering over temporal data: A survey (invited talk). In: Schewe, S., Schneider, T., Wijsen, J. (eds.) 24th International Symposium on Temporal Representation and Reasoning, TIME 2017, October 16-18, 2017, Mons, Belgium. LIPIcs, vol. 90, pp. 1:1–1:37. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2017). https://doi.org/10.4230/LIPIcs.TIME.2017.1, `https://doi.org/10.4230/LIPIcs.TIME.2017.1`
4. Artale, A., Kontchakov, R., Kovtunova, A., Ryzhikov, V., Wolter, F., Zakharyaschev, M.: First-order rewritability and complexity of two-dimensional temporal ontology-mediated queries. Journal of Artificial Intelligence Research **75**, 1223–1291 (2022). https://doi.org/10.1613/jair.1.13511, `https://doi.org/10.1613/jair.1.13511`
5. Artale, A., Kontchakov, R., Ryzhikov, V., Zakharyaschev, M.: A cookbook for temporal conceptual data modelling with description logics. ACM Transactions on Computational Logic **15**(3), 25:1–25:50 (2014). https://doi.org/10.1145/2629565, `https://doi.org/10.1145/2629565`
6. Artale, A., Mazzullo, A., Ozaki, A.: Temporal description logics over finite traces. In: Ortiz, M., Schneider, T. (eds.) Proceedings of the 31st International Workshop on Description Logics co-located with 16th International Conference on Principles of Knowledge Representation and Reasoning (KR 2018), Tempe, Arizona, US, October. CEUR Workshop Proceedings, vol. 2211. CEUR-WS.org (2018)
7. ASAM e.V.: Openxontology user guide 1.0.0 (01 2022), `https://www.asam.net/standards/asam-openxontology/`, hoehenkirchen, Germany. Standard
8. Baader, F., Borgwardt, S., Lippmann, M.: Temporalizing ontology-based data access. In: Bonacina, M.P. (ed.) Automated Deduction – CADE-24. pp. 330–344. Springer, Berlin, Germany (2013)
9. Baader, F., Borgwardt, S., Lippmann, M.: Temporal query entailment in the description logic shq. Journal of Web Semantics **33**, 71–93 (2015). https://doi.org/10.1016/j.websem.2014.11.008
10. Baader, F., Calvanese, D., McGuinness, D., Patel-Schneider, P., Nardi, D.: The description logic handbook: Theory, implementation and applications. Cambridge University Press (2003)
11. Babisch, S., Neurohr, C., Westhofen, L., Schoenawa, S., Liers, H., et al.: Leveraging the gidas database for the criticality analysis of automated driving systems. Journal of Advanced Transportation **2023** (2023)
12. Borgwardt, S., Lippmann, M., Thost, V.: Temporal query answering in the description logic dl-lite. In: Fontaine, P., Ringeissen, C., Schmidt, R.A. (eds.) Frontiers of Combining Systems. pp. 165–180. Springer, Berlin, Germany (2013)
13. Borgwardt, S., Thost, V.: Temporal query answering in the description logic el. In: Proceedings of the 24th International Conference on Artificial Intelligence. pp. 2819–2825. AAAI Press, Palo Alto, USA (2015)
14. Brandt, S., Kalaycı, E.G., Kontchakov, R., Ryzhikov, V., Xiao, G., Zakharyaschev, M.: Ontology-based data access with a horn fragment of metric temporal logic. In:

Proceedings of the AAAI Conference on Artificial Intelligence. vol. 31. AAAI Press, Palo Alto, USA (2017)

15. Chomicki, J.: Polynomial time query processing in temporal deductive databases. In: Rosenkrantz, D.J., Sagiv, Y. (eds.) Proceedings of the Ninth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems. pp. 379–391. ACM Press, New York, USA (1990). https://doi.org/10.1145/298514.298589, https://doi.org/10.1145/298514.298589

16. De Gelder, E., Manders, J., Grappiolo, C., Paardekooper, J.P., Den Camp, O.O., De Schutter, B.: Real-world scenario mining for the assessment of automated vehicles. In: 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC). pp. 1–8. IEEE, New York, USA (2020)

17. De Giacomo, G., Vardi, M.Y.: Linear temporal logic and linear dynamic logic on finite traces. In: IJCAI'13 Proceedings of the Twenty-Third international joint conference on Artificial Intelligence. pp. 854–860. AAAI Press, Palo Alto, USA (2013)

18. Elspas, P., Langner, J., Aydinbas, M., Bach, J., Sax, E.: Leveraging regular expressions for flexible scenario detection in recorded driving data. In: 2020 IEEE International Symposium on Systems Engineering (ISSE). pp. 1–8. IEEE, New York USA (2020)

19. Esterle, K., Gressenbuch, L., Knoll, A.: Formalizing traffic rules for machine interpretability. In: 2020 IEEE 3rd Connected and Automated Vehicles Symposium (CAVS). pp. 1–7. IEEE (2020)

20. Giacomo, G.D., Favorito, M.: Compositional approach to translate ltlf/ldlf into deterministic finite automata. In: Biundo, S., Do, M., Goldman, R., Katz, M., Yang, Q., Zhuo, H.H. (eds.) Proceedings of the Thirty-First International Conference on Automated Planning and Scheduling. pp. 122–130. AAAI Press, Palo Alto, USA (2021)

21. Guo, Y., Pan, Z., Heflin, J.: Lubm: A benchmark for owl knowledge base systems. Journal of Web Semantics **3**(2), 158–182 (2005)

22. Gutiérrez-Basulto, V., Jung, J.C., Kontchakov, R.: Temporalized EL ontologies for accessing temporal data: Complexity of atomic queries. In: Kambhampati, S. (ed.) Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence. pp. 1102–1108. IJCAI/AAAI Press, Palo Alto, USA (2016)

23. Gutiérrez-Basulto, V., Jung, J.C., Ozaki, A.: On metric temporal description logics. In: Kaminka, G.A., Fox, M., Bouquet, P., Hüllermeier, E., Dignum, V., Dignum, F., van Harmelen, F. (eds.) ECAI 2016 - 22nd European Conference on Artificial Intelligence. Frontiers in Artificial Intelligence and Applications, vol. 285, pp. 837–845. IOS Press, Amsterdam, The Netherlands (2016). https://doi.org/10.3233/978-1-61499-672-9-837

24. Horrocks, I., Kutz, O., Sattler, U.: The irresistible SRIQ. In: Grau, B.C., Horrocks, I., Parsia, B., Patel-Schneider, P.F. (eds.) Proceedings of the OWLED*05 Workshop on OWL: Experiences and Directions, Galway, Ireland, November 11-12, 2005. CEUR Workshop Proceedings, vol. 188. CEUR-WS.org (2005)

25. Horrocks, I., Tessaris, S.: A conjunctive query language for description logic aboxes. In: Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence. pp. 399–404. AAAI Press, Palo Alto, USA (2000)

26. Hülnhagen, T., Dengler, I., Tamke, A., Dang, T., Breuel, G.: Maneuver recognition using probabilistic finite-state machines and fuzzy logic. In: 2010 IEEE Intelligent Vehicles Symposium. pp. 65–70. IEEE, New York, USA (2010)

27. Hummel, B.: Description logic for scene understanding at the example of urban road intersections. Ph.D. thesis, Universität Karlsruhe (TH) (2009)

28. Isberner, M., Howar, F., Steffen, B.: The open-source learnlib. In: Kroening, D., Păsăreanu, C.S. (eds.) Computer Aided Verification. pp. 487–495. Springer, Berlin, Germany (2015)

29. Li, J., Vardi, M.Y., Rozier, K.Y.: Satisfiability checking for mission-time ltl. In: Dillig, I., Tasiran, S. (eds.) Computer Aided Verification. pp. 3–22. Springer, Berlin, Germany (2019)

30. Lippmann, M.: Temporalised description logics for monitoring partially observable events. Ph.D. thesis, Dresden University of Technology (2014)

31. Lucchetti, A., Ongini, C., Formentin, S., Savaresi, S.M., Del Re, L.: Automatic recognition of driving scenarios for adas design. In: IFAC Proceedings Volumes, Symposium on Advances in Automotive Control. vol. 49, pp. 109–114. Elsevier, Amsterdam, The Netherlands (2016)

32. Lutz, C., Wolter, F., Zakharyaschev, M.: Temporal description logics: A survey. In: 2008 15th International Symposium on Temporal Representation and Reasoning. pp. 3–14. IEEE, New York, USA (2008)

33. Maierhofer, S., Rettinger, A.K., Mayer, E.C., Althoff, M.: Formalization of interstate traffic rules in temporal logic. In: 2020 IEEE Intelligent Vehicles Symposium (IV). pp. 752–759. IEEE, New York, USA (2020)

34. Neurohr, C., Westhofen, L., Henning, T., de Graaff, T., Möhlmann, E., Böde, E.: Fundamental Considerations around Scenario-Based Testing for Automated Driving. In: 2020 IEEE Intelligent Vehicles Symposium. pp. 121–127. IEEE, New York, USA (2020). https://doi.org/10.1109/IV47402.2020.9304823

35. SAE International: J3016: Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles (2021), Warrendale, USA. Standard

36. Sirin, E., Parsia, B.: Optimizations for answering conjunctive abox queries: First results. In: Proc. of the 2006 Int. Workshop on Description Logics (DL'06). pp. 215–222 (2006)

37. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A practical owl-dl reasoner. Journal of Web Semantics **5**(2), 51–53 (2007)

38. Thost, V., Holste, J., Özçep, Ö.: On implementing temporal query answering in dl-lite. In: Proc. of the 28th Int. Workshop on Description Logics (DL'15). vol. 1350, pp. 552–555 (2015)

39. Wałega, P.A., Grau, B.C., Kaminski, M., Kostylev, E.V.: Datalogmtl over the integer timeline. In: Calvanese, D., Erdem, E., Thielscher, M. (eds.) Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning. pp. 768–777. IJCAI/AAAI Press, Palo Alto, USA (2020). https://doi.org/10.24963/kr.2020/79, https://doi.org/10.24963/kr.2020/79

40. Wang, D., Hu, P., Wałega, P.A., Grau, B.C.: Meteor: practical reasoning in datalog with metric temporal operators. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 36, pp. 5906–5913. AAAI Press, Palo Alto, USA (2022)

41. Westhofen, L.: Openllet Temporal Query Benchmarks (Dec 2023). https://doi.org/10.5281/zenodo.10436065, https://doi.org/10.5281/zenodo.10436065

42. Westhofen, L., Neurohr, C., Butz, M., Scholtes, M., Schuldes, M.: Using ontologies for the formalization and recognition of criticality for automated driving. IEEE Open Journal of Intelligent Transportation Systems **3**, 519–538 (2022)

43. Westhofen, L., Stierand, I., Becker, J.S., Möhlmann, E., Hagemann, W.: Towards a congruent interpretation of traffic rules for automated driving-experiences and challenges. In: Proceedings of the International Workshop on Methodologies for Translating Legal Norms into Formal Representations (LN2FR 2022) in association with the 35th International Conference on Legal Knowledge and Information Systems (JURIX 2022). pp. 8–21 (2022)
44. Zipfl, M., Koch, N., Zöllner, J.M.: A comprehensive review on ontologies for scenario-based testing in the context of autonomous driving. In: 2023 IEEE Intelligent Vehicles Symposium. pp. 1–7. IEEE, New York, USA (2023). https://doi.org/10.1109/IV55152.2023.10186681