Towards Computing Pointwise Repairs in a Fragment of DatalogMTL

Atefe Khodadaditaghanaki[1]

[1]Paderborn University, Paderborn, Germany

Abstract

When using datalogMTL over a dense timeline and inconsistent data, there can be several kinds of repairs as recently defined in [1]. One of such kinds is pointwise repairs for which no method for deciding the existence is known. In this paper, we consider datalogMTL$^{\diamondsuit}$, a fragment of datalogMTL, which has $\diamondsuit$ as the only temporal operator and a bounded dataset. We develop a computation algorithm that returns pointwise repairs for this setting.

1. Introduction

Metric temporal logic (MTL) was initially developed for modeling and reasoning about real-time systems [2]. MTL features two alternative semantics: *pointwise* and *continuous*. Typically, timestamps in both semantics come from a dense timeline $(\mathbb{T}, \leq)$, such as the Rationals. Using the continuous semantics, an interpretation assigns a set of propositional variables to each time point $t \in \mathbb{T}$. Significant research has been conducted in this area, leading to various approaches for handling temporal data [3]. A notable approach is datalogMTL, which combines metric temporal logic with Datalog [2]. A *datalogMTL* ontology consists of a set of rules (called a *program*) and data (called a *dataset*).

A significant practical challenge here is managing situations where the dataset is inconsistent with the ontology. While it is generally assumed that the program has been thoroughly debugged, the dataset often is large and subject to frequent changes, increasing the likelihood of errors. This challenge highlights the need for inconsistency-tolerant semantics [4], which allow for meaningful query answers despite inconsistencies in the ontology. A prominent form of inconsistency-tolerant semantics in the atemporal setting is repair-based semantics, which considers *subset repairs* of the ontology. Subset repairs are maximal subsets of the dataset consistent with the program. In our case, they restore consistent versions of the ontology.

Although there has been extensive research on handling inconsistencies and temporal data separately, addressing inconsistencies within a temporal knowledge bases has only been little explored so far, e.g. in [5, 1]. For datalogMTL over a dense timeline, there are several kinds of repairs defined in [1]. One of them are *pointwise* repairs. In general, a datalogMTL dataset consists of facts of the form $P(\vec{c})@\iota$, which expresses that the predicate P holds for the tuple of constants $\vec{c}$ at the interval $\iota \subseteq \mathbb{T}$. To introduce a pointwise repair of a dataset $\mathcal{D}$, we consider every fact $P(\vec{c})@\iota \in \mathcal{D}$ as $P(\vec{c})@\{t\}$ for all $t \in \iota$ to obtain another set of facts $\mathcal{D}'$. Then a pointwise repair of $\mathcal{D}$ is a subset repair of $\mathcal{D}'$. A crucial and basic question in this case is the existence of pointwise repairs. There are cases of datalogMTL ontologies that do not have any pointwise repair. Also, no techniques have been developed for computing a pointwise repair, if it exists. As a starting point to address this problem, we study *datalogMTL*$^{\diamondsuit}$, a fragment of datalogMTL and bounded datasets. A datalogMTL$^{\diamondsuit}$ program only contains $\diamondsuit$ as a temporal operator, whereas all the intervals are bounded in a bounded dataset. Our approach is to devise an algorithm that finds at least one pointwise repair in this setting. In [6], they introduced the ruler construction to essentially transform a continuous timeline to a discrete one, and it is easier to work in a discrete setting. We adapt this idea to achieve intervals of uniform length.

Logical Approaches to Handling Inconsistent Data (LINDA), July 24th, 2026, Lisbon

atefe@mail.uni-paderborn.de (A. Khodadaditaghanaki)

2. Preliminaries

Due to space limitations, we need to point the reader to an introduction to the technical notions given in [1]. We merely define the most relevant notions here:

Definition 2.1. A (possibly infinite) set of facts $\mathcal{D}$ is *pointwise included* (denoted $\mathcal{D} \sqsubseteq \mathcal{D}'$) in a set $\mathcal{D}'$, if for all $P(\vec{c})@\iota \in \mathcal{D}$ and all $t \in \iota$, there exists $P(\vec{c})@\iota' \in \mathcal{D}'$, s.t. $t \in \iota'$. Similarly, a fact $P(\vec{c})@\iota$ is a *pointwise member* of a set of facts $\mathcal{D}$ (denoted $P(\vec{c})@\iota \Subset \mathcal{D}$), if for all $t \in \iota$, there exists $P(\vec{c})@\iota' \in \mathcal{D}$, s.t. $t \in \iota'$. A *pointwise repair* $\mathcal{R}$ of a dataset $\mathcal{D}$ w.r.t. a program Π is a Π-consistent set of facts (i.e. $\mathcal{R}$ and Π are consistent) that is maximally pointwise included in $\mathcal{D}$. We use $pRep(\mathcal{D}, \Pi)$ to denote the set of pointwise repairs of $\mathcal{D}$ w.r.t. Π.

Definition 2.2. A dataset is *bounded* if all its intervals are bounded. The *set difference* between $\mathcal{D}$ and $\mathcal{D}'$ (denoted $\mathcal{D} \dot{-} \mathcal{D}'$) is a set $\mathcal{E}' \sqsubseteq \mathcal{D}$, s.t. $\mathcal{D}' \sqcap \mathcal{E}' = \varnothing$ and $\mathcal{E}$ is maximally pointwise included in $\mathcal{D}$.

A key technique for working with MTL over a dense timeline is to discretise the data, i.e. to identify intervals in which no changes occur and then use $\mathbb{Z}$ as a timeline. This is achieved by a ruler that is, in our case, determined by the data and the program.

Definition 2.3. Let Π be a program, $\mathcal{D}$ a dataset and $\mathrm{gcd}(\Pi, \mathcal{D})$ the greatest common divisor of all $t \in \mathbb{T}$, s.t. $\langle t, t' \rangle$, $\langle t', t \rangle$ or $\{t\}$ occurs in Π or $\mathcal{D}$ for some $t' \in \mathbb{T}$. A $(\Pi, \mathcal{D})$-*ruler* is the set of time points of the form $n \cdot \mathrm{gcd}(\Pi, \mathcal{D})$ for $n \in \mathbb{Z}$. A $(\Pi, \mathcal{D})$-*interval* is either a punctual interval $\{t\} = [t, t]$, where t is a time point in the ruler, or an interval (t_1, t_2) with t_1 and t_2 consecutive time points in the ruler. A $(\Pi, \mathcal{D})$-*dataset* $\mathcal{D}'$ is a dataset in which every interval is a $(\Pi, \mathcal{D})$-interval. For every datalogMTL program Π and bounded dataset $\mathcal{D}$, there is a *unique* $(\Pi, \mathcal{D})$-dataset $\mathcal{D}'$ s.t. $\mathcal{D} \sqsubseteq \mathcal{D}'$ and $\mathcal{D}' \sqsubseteq \mathcal{D}$. We call such a $\mathcal{D}'$ the $(\Pi, \mathcal{D})$-*dataset equivalent to* $\mathcal{D}$.

Observe that the construction of a $(\Pi, \mathcal{D})$-ruler guarantees that (i) the $(\Pi, \mathcal{D})$-intervals have uniform length and that (ii) within a $(\Pi, \mathcal{D})$-interval, no changes in the extensions of predicates occur.

3. Towards pointwise repairs for DatalogMTL$^{\diamondsuit}$

We start by considering full datalogMTL and illustrate that there are some datalogMTL ontologies that do not have any pointwise repairs.

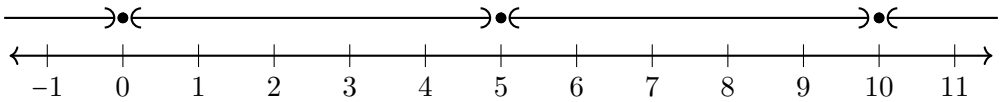
Example 3.1. *Let* $\Pi_1 = \{\bot \leftarrow \boxplus_{(0,\infty)} P\}$ *and* $\mathcal{D}_1 = \{P@(0, \infty)\}$. Π_1 *expresses that if* P *holds at some time point until infinity, then there is an inconsistency.* $\mathcal{D}_1$ *states that* P *holds at* 0 *until infinity. So,* Π_1 *and* $\mathcal{D}_1$ *are inconsistent. If* $\mathcal{R}$ *is a pointwise repair of* $\mathcal{D}_1$ *w.r.t.* Π_1*, then there are two cases:*

1. *There are two time points* $t_1 < t_2$ *s.t.* $P@\{t_1\}, P@\{t_2\} \Subset \mathcal{D}_1 \dot{-} \mathcal{R}$*. Then,* $\mathcal{R}' = \mathcal{R} \cup \{P@\{t_1\}\}$ *is also* Π_1*-consistent. This is a contradiction with* $\mathcal{R}$ *being pointwise inclusion-maximal.*
2. *There is only one time point* t *s.t.* $P@\{t\} \Subset \mathcal{D}_1 \dot{-} \mathcal{R}$*. Then,* $P@(t, \infty) \Subset \mathcal{R}$*, thus* $\boxplus_{(0,\infty)} P$ *holds at* t*, but so does* $\bot$*, which is a contradiction with* Π_1*-consistency of* $\mathcal{R}$*.*

So, there is no pointwise repair of $\mathcal{D}_1$ *w.r.t.* Π_1*.*

Now, if we consider datalogMTL$^{\diamondsuit}$ and use a bounded dataset, then a pointwise repair always exists (as we will show). The next examples and their pointwise repairs illustrate already ideas underlying our algorithm for computing pointwise repairs.

Example 3.2. *Let* $\Pi_2 = \{\bot \leftarrow P \wedge \diamondsuit_{(0,5)} P\}$ *and* $\mathcal{D}_2 = \{P@[0, 10]\}$. Π_2 *espresses that* P *cannot hold the two timepoints* $t_1 < t_2$ *at the same time, where* $t_2 - t_1 < 0.5$*. Please observe that we are using different kinds of intervals (open and closed). The* $(\Pi_2, \mathcal{D}_2)$*-ruler is:*



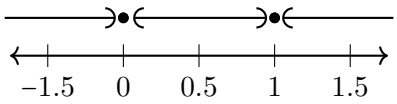
So, the $(\Pi_2, \mathcal{D}_2)$-dataset equivalent to $\mathcal{D}_2$ is $\mathcal{D}_2' = \{P@\{0\},\ P@(0,5),\ P@\{5\},\ P@(5,10),\ P@\{10\}\}$. If we add the elements of $\mathcal{D}_2'$ one by one in the order listed to an initially empty set $\mathcal{R}$ so that it remains Π_2-consistent, we obtain $\mathcal{R} = \{P@\{0\},\ P@\{5\},\ P@\{10\}\}$. Since $P@\{0\}$ and $P@(0,5)$ cause inconsistency, we cannot add $P@(0,5)$ to $\mathcal{R}$. The same thing holds for $P@\{5\}$ and $P@(5,10)$. To make $\mathcal{R}$ pointwise inclusion maximal, we need to check whether we can add more facts to it. Suppose that we add $P@\{t\}$ to $\mathcal{R}$, where w.l.o.g. $0 < t < 5$, then $P \wedge \diamondminus_{(0,5)} P$ would hold at 0. The same holds for $5 < t < 10$, which makes $P \wedge \diamondminus_{(0,5)} P$ hold at time point 5. So, there is no other fact in $\mathcal{D}_2$ that can be added to $\mathcal{R}$ so that it remains Π_2-consistent. Therefore, $\mathcal{R}$ is a pointwise repair of $\mathcal{D}_2$ w.r.t. Π_2.

This is an easy case, where a pointwise repair exists already w.r.t. the initial ruler. We now turn to a more intricate case, where, depending on the partial repair, the ruler needs to be recalibrated, i.e., adapted to a new base unit.

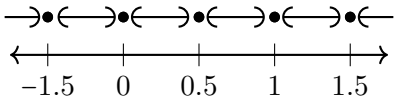
Example 3.3. *Let the program Π_3 and dataset $\mathcal{D}_3$ be of the following form:*

$$\begin{aligned}\Pi_3 = \{&Q \leftarrow \diamondminus_{(6,9)} P,\ U \leftarrow \diamondminus_{(4,6)} P,\ T \leftarrow \diamondminus_{(5,6)} P,\ \bot \leftarrow Q \wedge U,\ \bot \leftarrow Q \wedge T,\\ &\bot \leftarrow P \wedge \diamondminus_{\{4\}} P,\ \bot \leftarrow P \wedge \diamondminus_{(1,3)} P\},\\ \mathcal{D}_3 = \{&P@(0,1),\ P@(2,3),\ P@(4,5)\}.\end{aligned}$$

Then the $(\Pi_3, \mathcal{D}_3)$- ruler is:



Here, the $(\Pi_3, \mathcal{D}_3)$-dataset equivalent to $\mathcal{D}_3$ is $\mathcal{D}_3$ itself. If we try to add all the elements of $\mathcal{D}_3$ one by one in the given order to an empty set $\mathcal{R}$, we see that nothing can be added and preserve Π_3-consistency. Because adding $P@(0,1)$ makes Q hold at $(6,10)$, U at $(4,7)$, and T at $(5,7)$ which makes $\mathcal{R}$ and Π_3 inconsistent. The same argument holds for $P@(2,3)$ and $P@(4,5)$. Now, we need to see if we can add at least a part of these facts to $\mathcal{R}$ to make $\mathcal{R}$ pointwise inclusion maximal. By adding $P@\{0.5\}$ (a part of $P@(0,1)$), we make Q hold at $(6.5, 9.5)$, U at $(4.5, 6.5)$, and T at $(5.5, 6.5)$, which does not cause any inconsistencies (we could add any $P@\{t\}$ for $0 < t < 1$ instead of $P@\{0.5\}$). But once $\mathcal{R} = \{P@\{0.5\}\}$, then adding any other fact of the form $P@\{t\}$ for $0 < t < 1$ and $t \neq 0.5$ to $\mathcal{R}$ makes Π_3 and $\mathcal{R}$ inconsistent, and the reason is similar to what we had above. Now, for Π_3 with $\mathcal{R}$ we obtain the new $(\Pi_3, \mathcal{R})$-ruler :



To extend $\mathcal{R}$ s.t. it is pointwise maximal included in $\mathcal{D}$, still, temporal subparts of facts contained in $\mathcal{D}_3$ can be added to $\mathcal{R}$. To this end, we need to consider the new ruler and the $(\Pi_3, \mathcal{R})$-dataset equivalent to $\mathcal{D}_3$ is

$$\mathcal{D}' = \{P@(2, 2.5),\ P@\{2.5\},\ P@(2.5, 3),\ P@(4, 4.5),\ P@\{4.5\},\ P@(4.5, 5)\}.$$

In order to be able to "trace" that the first three facts in $\mathcal{D}'$ originate from the original fact $P@(2,3) \in \mathcal{D}$ and the last three from the original fact $P@(4,5) \in \mathcal{D}$, we use the set of sets

$$S = \big\{\{P@(2, 2.5),\ P@\{2.5\},\ P@(2.5, 3)\},\ \{P@(4, 4.5),\ P@\{4.5\},\ P@(4.5, 5)\}\big\}$$

instead of $\mathcal{D}'$. None of the facts in $\mathcal{D}'$ (or $\bigcup S$ respectively) can be added to $\mathcal{R}$ as a whole, and the reason for the non-punctual ones is the same reason we gave for $P@(0,1)$. Adding $P@\{2.5\}$ to $\mathcal{R}$ results in an inconsistency due to the rule $\bot \leftarrow P \wedge \diamondminus_{(1,3)} P \in \Pi_3$ and the fact $P@(0.5) \in \mathcal{R}$. Also, adding $P@\{4.5\}$ to $\mathcal{R}$ results in an inconsistency due to the rule $\bot \leftarrow P \wedge \diamondminus_{\{4\}} P \in \Pi_3$ and the fact $P@(0.5) \in \mathcal{R}$. We also cannot add any fact of the form $P@\{t\}$, where $2 < t < 3$, because of the rule $\bot \leftarrow P \wedge \diamondminus_{(1,3)} P \in \Pi_3$. So, we

define Algorithm PointwiseRepair$^{\diamondsuit}(\Pi, \mathcal{D})$
Input: DatalogMTL$^{\diamondsuit}$ program Π and bounded dataset $\mathcal{D}$
Output: pointwise repair $\mathcal{R}$ of $\mathcal{D}$ w.r.t. Π

01 $\mathcal{D}' :=$ DatasetEquivalent$(\Pi, \mathcal{D}, \mathcal{D})$
02 $\mathcal{R} := \varnothing$ and $S := \varnothing$
03 **for** all $P(\vec{c})@\iota \in \mathcal{D}'$
04 **if** $\mathcal{R} \cup \{P(\vec{c})@\iota\}$ is Π-consistent, **then** $\mathcal{R} := \mathcal{R} \cup \{P(\vec{c})@\iota\}$
05 **else if** ι is not punctual, **then** $S := S \cup \big\{\{P(\vec{c})@\iota\}\big\}$
06 **repeat**
07 pick $\mathcal{T} \in S, P(\vec{c})@\iota \in \mathcal{T}$, and $t \in \iota$
08 $\mathcal{T} := \mathcal{T} \dot{-} \{P(\vec{c})@\iota\}$
09 **if** $\mathcal{T} = \varnothing$, **then** $S := S \dot{-} \{\mathcal{T}\}$
10 **if** $\mathcal{R} \cup \{P(\vec{c})@\{t\}\}$ is Π-consistent
11 $\mathcal{R} := \mathcal{R} \cup \{P(\vec{c})@\{t\}\}$
12 $S := S \dot{-} \{\mathcal{T}\}$
13 **if** $\bigcup S$ is not a $(\Pi, \mathcal{R})$-dataset
14 **for** all $\mathcal{T}' \in S$ let $\mathcal{T}' =$ DatasetEquivalent$(\Pi, \mathcal{R}, \mathcal{T}')$
15 **until** S is empty
16 **return** $\mathcal{R}$

Figure 1: The Computation Algorithm.

cannot add any part of facts $P@(2, 2.5)$ *and* $P@(2.5, 3)$ *to* $\mathcal{R}$*. Moreover, we cannot add any fact of the form* $P@\{t\}$*, where* $4 < t < 4.5$*, because it makes* U *hold at* $(t+4, t+6)$ *and* $8 < t+4 < 8.5$*. So, the interval intersects with* $(6.5, 9.5)$*, where* Q *holds, which raises an inconsistency due to the rule* $\bot \leftarrow Q \wedge U \in \Pi_3$*. Thus, we cannot add any part of fact* $P@(4, 4.5)$ *to* $\mathcal{R}$*. But, by the same reasoning we had for* $P@(0.5)$*, we can add any fact of the form* $P@\{t\}$ *to* $\mathcal{R}$*, where* $4.5 < t < 5$*, but only* one*. So finally, one of the pointwise repairs for the datalogMTL*$^{\diamondsuit}$ *ontology consisting of* Π_3 *and* $\mathcal{D}_3$ *is* $\mathcal{R} = \{P@\{0.5\},\ P@\{4.75\}\}$*.*

4. Computation algorithm for a single pointwise repair in DatalogMTL$^{\diamondsuit}$

We present an algorithm for computing a pointwise repair for any inconsistent datalogMTL$^{\diamondsuit}$ program Π and bounded dataset $\mathcal{D}$. The main idea is as shown in Example 3.3. First, the facts in $\mathcal{D}$ are split (in terms of their "duration") and make a new dataset $\mathcal{D}'$, which is the $(\Pi, \mathcal{D})$-dataset equivalent to $\mathcal{D}$. Then, the algorithm first tries to add facts from $\mathcal{D}'$ as a whole one by one to an initially empty set $\mathcal{R}$, as long as the set stays Π-consistent. For those facts $P(\vec{c})@\iota$ that cannot be added as a whole, a time point $t \in \iota$ is chosen. If $P(\vec{c})@\{t\}$ is consistent with Π and the current $\mathcal{R}$, it is added to $\mathcal{R}$. If $P(\vec{c})@\{t\}$ causes an inconsistency, it is skipped and $P(\vec{c})@\iota$ gets deleted from $\mathcal{D}'$. After each addition of such a temporal sub-fact as $P(\vec{c})@\{t\}$, the ruler needs to be recalibrated each time, and also the facts in $\mathcal{D}'$ get split accordingly, to make it a $(\Pi, \mathcal{R})$-dataset. The algorithm proceeds in this way for every fact in $\mathcal{D}'$ until $\mathcal{D}'$ is empty. It returns $\mathcal{R}$, which can either be a pointwise repair or the empty set.

The formal version of this idea is the algorithm PointwiseRepair$^{\diamondsuit}(\Pi, \mathcal{D})$ shown in Figure 1. It uses the procedure DatasetEquivalent$(\Pi, \mathcal{D}, \mathcal{D}')$ which computes the $(\Pi, \mathcal{D})$-dataset equivalent to $\mathcal{D}'$, i.e. recalibrates the ruler.

Theorem 4.1. *The algorithm PointwiseRepair*$^{\diamondsuit}(\Pi, \mathcal{D})$

1. *returns only* Π*-consistent data sets,*
2. *returns only data sets that are maximally pointwise included in* $\mathcal{D}$*, and*
3. *terminates in finite time.*

Proof. (sketch) The first claim is easy, because the algorithm never adds a fact to $\mathcal{R}$ which makes it Π-inconsistent due to the checks in lines 4 and 10. To show the other two statements, we need the following claim.

Claim 1: For every program Π, Π-consistent dataset $\mathcal{D}$ and non-punctual $(\Pi, \mathcal{D})$-interval ι it holds that, if $\mathcal{D} \cup \{P(\vec{c})@\iota\}$ is Π-inconsistent, and $\iota' \subset \iota$ is non-punctual, then $\mathcal{D} \cup \{P(\vec{c})@\iota'\}$ is also Π-inconsistent.

The intuition for Claim 1 to hold is that the cause in $P(\vec{c})@\iota$ that makes $\mathcal{D}$, Π-inconsistent, is also present in $P(\vec{c})@\iota'$. Claim 1 can be proved by induction on the rule applications which are used to make the Π-inconsistency with $\mathcal{D} \cup \{P(\vec{c})@\iota\}$ and show that the same results would be obtained for $\mathcal{D} \cup \{P(\vec{c})@\iota'\}$. Also, for two time points $t_1 \neq t_2 \in \iota$ we have: $\mathcal{D} \cup \{P(\vec{c})@\iota\}$ is Π-inconsistent iff $\mathcal{D} \cup \{P(\vec{c})@\{t_1\},\ P(\vec{c})@\{t_2\}\}$ is Π-inconsistent. We already saw in Example 3.3 that if ι' is punctual, then Claim 1 need not hold. That is the reason for the loop in line 6. Note that if we can add any fact of the form $P(\vec{c})@\{t\}$ to $\mathcal{R}$ where $t \in \iota$ s.t. $\mathcal{R}$ and Π remain consistent, then $\mathcal{R} \cup \{P(\vec{c})@\{t\}\}$ and Π are also consistent for any other $t' \in \iota$. Together with Claim 1, we have the pointwise inclusion maximality of PointwiseRepair$^{\diamondsuit}(\Pi, \mathcal{D})$, because if no fact $P(\vec{c})@\iota$ can be added to $\mathcal{R}$, then at most one part of it of the form $P(\vec{c})@\{t\}$ can be added to $\mathcal{R}$, where $t \in \iota$. This step is done in the main loop that starts in Line 6.

To prove that PointwiseRepair$^{\diamondsuit}(\Pi, \mathcal{D})$ terminates, we need to prove that the main loop terminates. This would happen if S becomes empty. To show this, it is sufficient to prove that the execution of the algorithm reaches Line 14 only finitely many times, since only in this line S gets increased. First, we need to see why the algorithm reaches Line 14. Intuitively, this line does what was done in Example 3.3 after adding $P@\{0.5\}$ to $\mathcal{R}$. There was a fact $P(\vec{c})@\iota \in \mathcal{D}'$ that we could not add to $\mathcal{R}$, but we could add a temporal sub-part of it to $\mathcal{R}$. As already mentioned, we cannot add two facts $P(\vec{c})@\{t_1\}$ and $P(\vec{c})@\{t_2\}$ to $\mathcal{R}$, where $t_1 \neq t_2 \in \iota$. Since $\mathcal{D}$ is bounded, $\mathcal{D}'$ is finite, and for every fact in $\mathcal{D}'$, we reach line 14 at most once. This proves that S increases only finitely many times. It is also easy to check that in every iteration of the main loop, the cardinality of $\bigcup S$ (w.r.t. the last ruler) decreases. Since S is finite, it empties after finitely many steps, and the algorithm terminates. □

Corollary 4.2. *PointwiseRepair*$^{\diamondsuit}(\Pi, \mathcal{D})$ *is*

- *sound, i.e. if it returns $\mathcal{R}$, then $\mathcal{R}$ is a pointwise repair of $\mathcal{D}$ w.r.t. Π.*
- *complete, i.e. it always returns a pointwise repair of $\mathcal{D}$ w.r.t. Π.*

Although the non-deterministic choice in Line 7 can, in principle, be used to get different pointwise repairs for the same ontology, it is not clear whether (and if so, how) the algorithm can be used to return all the pointwise repairs. Although PointwiseRepair$^{\diamondsuit}(\Pi, \mathcal{D})$ solves the open problem from [1], it cannot be employed to address query answering under pointwise repair semantics.

5. Conclusion and future work

In this paper, we addressed the problem of the existence of pointwise repairs for a fragment of datalogMTL and developed an algorithm to compute some of the pointwise repairs for datalogMTL$^{\diamondsuit}$. Due to space limitations, we only reported on some results regarding the temporal operator $\diamondsuit$. We also studied other fragments of datalogMTL over bounded datasets, such as datalogMTL$^{\boxminus}$ (which only has $\boxminus$ as a temporal operator), datalogMTL$^{\diamondsuit,\Box}$ (where the only temporal operators absent are $\mathcal{U}$ and $\mathcal{S}$), and bounded datalogMTL programs. In fact, in all three of these fragments, we have obtained algorithms to compute pointwise repairs. It is not clear yet how these methods can be combined into an algorithm for computing pointwise repairs for bigger fragments of datalogMTL.

As future work, we would like to see if the results for datalogMTL$^{\diamondsuit}$ can be transferred to the case of unbounded datasets. Moreover, it would be interesting to find a characterisation for the (non-)existence of pointwise repairs for datalogMTL ontologies with an unbounded dataset, as these are often used in practice. Such a characterisation would facilitate the use of repair semantics based on pointwise repairs for inconsistent datalogMTL ontologies. Also, computing the complexity of our algorithms is part of our future work.

References

[1] M. Bienvenu, C. Bourgaux, A. Khodadaditaghanaki, Inconsistency handling in datalogmtl, in: J. Kwok (Ed.), Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI-25, International Joint Conferences on Artificial Intelligence Organization, 2025, pp. 4374–4381. URL: https://doi.org/10.24963/ijcai.2025/487. doi:10.24963/ijcai.2025/487, main Track.

[2] S. Brandt, E. G. Kalayci, V. Ryzhikov, G. Xiao, M. Zakharyaschev, Querying log data with metric temporal logic, J. Artif. Intell. Res. 62 (2018) 829–877. URL: https://doi.org/10.1613/jair.1.11229. doi:10.1613/JAIR.1.11229.

[3] A. Artale, R. Kontchakov, A. Kovtunova, V. Ryzhikov, F. Wolter, M. Zakharyaschev, Ontology-mediated query answering over temporal data: A survey (invited talk), in: S. Schewe, T. Schneider, J. Wijsen (Eds.), 24th International Symposium on Temporal Representation and Reasoning, TIME 2017, October 16-18, 2017, Mons, Belgium, volume 90 of *LIPIcs*, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017, pp. 1:1–1:37. URL: https://doi.org/10.4230/LIPIcs.TIME.2017.1. doi:10.4230/LIPICS.TIME.2017.1.

[4] M. Bienvenu, C. Bourgaux, Inconsistency-tolerant querying of description logic knowledge bases, in: J. Z. Pan, D. Calvanese, T. Eiter, I. Horrocks, M. Kifer, F. Lin, Y. Zhao (Eds.), Reasoning Web: Logical Foundation of Knowledge Graph Construction and Query Answering - 12th International Summer School 2016, Aberdeen, UK, September 5-9, 2016, Tutorial Lectures, volume 9885 of *Lecture Notes in Computer Science*, Springer, 2016, pp. 156–202. URL: https://doi.org/10.1007/978-3-319-49493-7_5. doi:10.1007/978-3-319-49493-7_5.

[5] C. Bourgaux, P. Koopmann, A. Turhan, Ontology-mediated query answering over temporal and inconsistent data, Semantic Web 10 (2019) 475–521. URL: https://doi.org/10.3233/SW-180337. doi:10.3233/SW-180337.

[6] P. A. Walega, B. C. Grau, M. Kaminski, E. V. Kostylev, Tractable fragments of datalog with metric temporal operators, in: C. Bessiere (Ed.), Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020, ijcai.org, 2020, pp. 1919–1925. URL: https://doi.org/10.24963/ijcai.2020/266. doi:10.24963/IJCAI.2020/266.