Logical Data Expiration for Fixpoint Extensions of Temporal Logics

David Toman

School of Computer Science, University of Waterloo
Waterloo, Ontario, Canada N2L 3G1
`david@uwaterloo.ca`

Abstract. We study the differences between the future and the past fragments of fixpoint extensions of first-order temporal logics in their relationship to expiration of database histories. We show that while the past fragment admits a bounded expiration operator, the future one requires retaining data of size bounded from below by a function linear in the length of the history. We also discuss fragments of future fixpoint temporal logic for which bounded expiration operators can exist.

1 Introduction

Evolution of data is common to most database and information systems. While it is often sufficient to retain only the most current version of a database, many applications of information systems require maintaining information about the past states of the database as well, often in the form of a history of the evolution of the data set. This functionality might be required, among other uses, for archival, analytical, and/or legal reasons.

However, archiving past data indefinitely is costly and in many situations, it is not feasible to retain all past data. In addition, even if collecting an ever growing history of the past states of the database were possible, querying such large data sets might become prohibitively expensive. To address these problems, various techniques have been proposed to identify parts of a history that are no longer needed and subsequently removing such parts. These techniques range from ad-hoc administrative approaches [10, 16] to logical, query-driven expiration algorithms [2, 20]. These algorithms are based, in general, on analyzing queries posed over a history to determine which parts of the current history can be removed without affecting answers of these queries with respect to the current history and all possible extension of the current history. A recent survey chapter [21] discusses many of these approaches and the state of art in the field in detail.

In this paper, we study query-driven approaches to data expiration in the context of *fixpoint extensions of first-order temporal logic*. The main results of the paper are as follows.

1. We present a technique that, for (a fixed set of) queries formulated in the past fragment of fixpoint first-order temporal logic, allows substantial amounts of historical data to expire, while preserving answers to the given queries.

In particular, we show that the amount of data that has to be retained is independent of the length of the history and depends only on the size of the active data domain and the query.

2. We show that such a technique cannot be developed for the future fragment. Moreover, we show that there are queries in this fragment for which the size of the retained data is bounded from below by a function linear in the length of the history.

The results demonstrate a large gap between the behavior of the past and future fragments of fixpoint first-order temporal logics with respect to data expiration. The gap can be traced to the interaction between (data) quantifiers and fixpoint operators. In particular, and despite of the popularity of the future fragment of *propositional* temporal logic in many other areas of computer science [22–24], the future fragment, as a query language over database histories, exhibits a severe deficiency that relates to the ability to expire past data, when compared to the past fragment.

The paper is organized as follows: Section 2 provides the necessary background definitions. Section 3 discusses related approaches. Section 4 defines a bounded expiration operator for past fixpoint temporal logic queries. Section 5 shows that expiration operators for future fixpoint temporal logic queries require linear space (in the length of the database history) to store the residual data. Section 6 concludes with directions for future work.

2 Definitions

In this section we formally introduce database histories, fixpoint extensions of temporal logic as a query language over such histories, and a basic framework in which data expiration can be studied.

2.1 Temporal Databases and Fixpoint Temporal Logics

Temporal databases provide a uniform and systematic way of dealing with historical data [5]. For the purposes of this paper we only consider *transaction-time* temporal databases—temporal databases that collect information about modifications of a standard relational database in a form of a *history*.

Intuitively, a relational database history records the facts that the instance of the database at time i was D_i in a (time-instant indexed) sequence of successive database instances (states of the history). A linear order of the sequence is implied by the natural ordering of the time instants with respect to the flow of time. The discrete and linearly ordered nature of the set of time instants—the temporal domain—also yields the notion of previous and next states in the database history in terms of consecutive time instants. Therefore we assume, without loss of generality, that time is modeled by natural numbers with the usual linear order. In particular natural numbers will be used to index the individual states in histories. From a temporal database point of view, the histories can be viewed as append-only, transaction-time temporal databases [5, 17].

Definition 1 (History) *Let ρ be a relational signature. A* history *H is a finite integer-indexed sequence of databases*

$$H = \langle D_0, D_1, \ldots, D_n \rangle$$

where D_i is a standard relational database over ρ. We call D_i a state *of H at the* time instant *i.*

The data domain *$\mathbf{D}_H$ of H is the union of all (data) values that appear in any relation in D_i at any time instant i; the* temporal domain *$\mathbf{T}_H$ is the set of all time instants that appear as indices in the history H.*

Histories allow us to formulate queries that span multiple database states. To facilitate querying, many *temporal query languages* have been developed [5, 17–19]. Among the more popular choices, temporal logic based query languages play a significant role, mainly due to their ease of use, well-defined semantics, amenability to implementation, and well-studied theoretical properties.

In this paper we focus on μTL, a fixpoint extension of first-order temporal logic (FOTL). The extension is realized by extending the syntax with a *fixpoint operator* of the form $\mu X.Q$ where X is a *relational variable* standing for a (new) predicate symbol over the free variables of Q. The intuitive meaning of such an operator is the (least) solution to the $X \equiv Q$ sentence. The following definition gives the syntax of μTL used in this paper.

Definition 2 (Syntax of μTL) *A μTL* formula (or query) *is an expression in the grammar*

$$Q ::= r(x_1, \ldots, x_k) \mid x_i = x_j \mid Q \wedge Q \mid \neg Q \mid \exists x.Q \mid \bullet Q \mid \circ Q \mid \mu X.Q.$$

where $x_1, \ldots, x_k$ are (data) variable names and r and X predicate symbols. The $\bullet$ and $\circ$ temporal connectives stand for the "previous time" and "next time" modalities, respectively. We also assume that each occurrence of the $\mu X.Q$ subformula in a μTL formula uses a distinct symbol X.

We say that a μTL formula is guarded *if for every subformula of the form $\mu X.Q$ all occurrences of X in Q are in the scope of one or more occurrences of either the $\bullet$ or the $\circ$ connectives (but not both).*

We say that a μTL formula is well-formed *if for every subformula of the form $\mu X.Q$ all occurrences of X in Q are guarded or positive. We also require data variables in well-formed formulas to be range-restricted [1, 6].*

We define the past fragment of μTL to be the well formed μTL formulas that use only the $\bullet$ connective (we denote this fragment by PastμTL*), and the future fragment, that uses only the $\circ$ connective (denoted by* FutureμTL*).*

In the rest of this paper we consider only well-formed μTL formulas in one of the above fragments. The well-formedness requirement guarantees unique answers to μTL queries over *finite histories*[1].

Definition 3 (Semantics of well-formed μTL formulas) *Let H be a finite history $\langle D_0, D_1, \ldots, D_n \rangle$, θ a substitution of values from $\mathbf{D}_H$ for variables, and $t \in \mathbf{T}_H$ a time instant. The semantics of μTL is defined as follows*

$$
\begin{array}{ll}
H, \theta, t \models r(x_1, \ldots, x_k) & \text{if } (x_1\theta, \ldots, x_k\theta) \in R^t \\
H, \theta, t \models x_i = x_j & \text{if } \theta(x_i) = \theta(x_j) \\
H, \theta, t \models Q_1 \wedge Q_2 & \text{if } H, \theta, t \models Q_1 \text{ and } H, \theta, t \models Q_2 \\
H, \theta, t \models \neg Q & \text{if not } H, \theta, t \models Q \\
H, \theta, t \models \exists x_i . Q & \text{if there is } a \in \mathbf{D}_H \text{ such that } H, \theta[a/x_i], t \models Q \\
H, \theta, t \models \bullet Q & \text{if } t-1 \in \mathbf{T}_H \text{ and } H, \theta, t-1 \models Q \\
H, \theta, t \models \bigcirc Q & \text{if } t+1 \in \mathbf{T}_H \text{ and } H, \theta, t+1 \models Q \\
H, \theta, t \models \mu X . Q & \text{if } H, \theta, t \models Q^{\omega}[\emptyset/X]
\end{array}
$$

where R^t is the instance of r in D_t and Q^{ω} is the countably infinite unfolding of the fixpoint operator[2].

An answer to a FutureμTL *query Q with free variables $x_1, \ldots, x_k$ with respect to a history H is the relation*

$$Q(H) = \{(a_1, \ldots, a_k) \in \mathbf{D}_H^k : H, [a_1/x_1, \ldots, a_k/x_k], 0 \models Q\},$$

where 0 is the time instant associated with the first state in H. Similarly, an answer to a PastμTL *query Q with free variables $x_1, \ldots, x_k$ with respect to the history H is the relation*

$$Q(H) = \{(a_1, \ldots, a_k) \in \mathbf{D}_H^k : H, [a_1/x_1, \ldots, a_k/x_k], n \models Q\},$$

where n is the time instant associated with the last state in H.

Note that the temporal variables used in the semantic definitions are *restricted to the temporal domain of the history $\mathbf{T}_H$*. Thus, for finite histories, $\bullet Q$ is false in 0-th state of H independently of Q. Similarly, $\bigcirc Q$ is false in the last state of the history[3].

Standard first-order temporal operators can be defined in the appropriate μTL fragment.

---

[1] For infinite histories or when mixed guards are allowed we have to require guarded occurrences of X to be positive as well in order to guarantee uniqueness of least fixpoints and in turn of query answers.

[2] Since the history is finite, there will always be a unique fixpoint that can be reached after finitely many steps of unfolding of Q.

[3] We briefly discuss alternatives to this definition, including the consideration of possibly infinite histories, in Section 3.

Example 4 The classical **since** and **until** connectives can be defined (assuming discrete time domain) by the following guarded (and positive) fixpoint formulas:

$$\varphi\ \textbf{since}\ \psi = \mu X.\varphi \wedge \bullet(\psi \vee X),$$
$$\varphi\ \textbf{until}\ \psi = \mu X.\varphi \wedge \circ(\psi \vee X);$$

this definition exactly mirrors the standard *unfolding rule* for these connectives (we are assuming for simplicity that φ and ψ have the same sets of free data variables; otherwise we could simply use standard equivalences to remedy this situation). Additional connectives, e.g., the $\Box$ ("always in the future"), $\Diamond$ ("eventually in the future"), $\blacksquare$ ("always in the past"), and $\blacklozenge$ ("sometime in the past") connectives can be defined in terms of **since** and **until** [7].

2.2 Expiration Operators

Since database histories grow over time, a clear approach to controlling the space needed to store the history is an important issue. These approaches range from administrative tools for discarding obsolete parts of the history to formal, query-driven approaches—those that only discard data that can be shown not to be needed subsequently for answering (a fixed set of) queries over the history. The following definitions follow the development in [21], a recent survey of data expiration for historical databases.

Definition 5 (Expiration Operator)
Let $\mathcal{L}$ be a temporal query language over database histories and $Q \in \mathcal{L}$ a query in the language. We define an expiration operator *for Q as a triple*

$$\mathcal{E}(Q) = (0^{\mathcal{E}}, \Delta^{\mathcal{E}}, Q^{\mathcal{E}}).$$

The first two components, $0^{\mathcal{E}}$ and $\Delta^{\mathcal{E}}$, provide an inductive definition of the actual expiration operator as follows:

$$\begin{array}{rll} \mathcal{E}(\langle\,\rangle) = 0^{\mathcal{E}} & & \text{(initial state)},\\ \mathcal{E}(H;D) = \Delta^{\mathcal{E}}(\mathcal{E}(H),D) & & \text{(extension maintenance)}, \end{array}$$

where $\langle\,\rangle$ is the empty history and $H;D$ is the extension of a history H by an additional state D. The result of the (inductive) application of these functions define the result of the expiration process with respect to extensions of a history. We call the result $\mathcal{E}(H)$ the residual history.

The third component, $Q^{\mathcal{E}}$, is a query over the residual history that mimics the original query Q. The three components must maintain the following soundness condition:

$$Q(H) = Q^{\mathcal{E}}(\mathcal{E}(H)) \qquad \text{(answer preservation)}.$$

In general, there is no restriction on the structure of $\mathcal{E}(H)$, in particular we do not insist that this value has to conform to a particular data model, e.g., the data models of H or D.

This definition allows us to study the properties of various expiration operators, in particular the space needed to represent the residual history $\mathcal{E}(H)$. The space can be measured in terms of the length of the history ($|\mathbf{T}_H|$), the size of the active data domain ($|\mathbf{D}_H|$), and the size of the query, $|Q|$. Of particular interest is the dependence of $|\mathcal{E}(H)|$ on $|\mathbf{T}_H|$, since the length of the history grows over time. We are especially interested in expiration operators that guarantee a *bounded history encoding* [2], an encoding for residual histories whose size does not depend on the length of the history but only on the sizes of the active data domain and the query. This measure is probably the most interesting one, since we assume that the query (and thus also its size) is fixed and we have no control over the size of the active data domain $\mathbf{D}_H$: a user transaction can append arbitrarily large *next* state to the current history.

3 Related Work

There are two main approaches to data expiration. The first approach is based on administrative rules and policies for identifying data to be expired and has been considered by Skyt, Jensen, *et al.* [10, 16].

The focus of this paper, however, is on logical query-driven data expiration and on the limits of such approaches for powerful temporal query languages. The first result in the paper extends Chomicki's approach [2, 4] to enforcing temporal integrity constraints formulated in past first-order temporal logic (PastTL) to queries formulated in *fixpoint extensions* of first-order temporal logic (μTL and, in particular its past fragment PastμTL), a language strictly more expressive than first-order temporal logic. Indeed, PastμTL allows formulation of queries (and properties) that cannot be formulated in PastTL [24], e.g., the "p is true in every even state" property.

The second, negative result on the future fragment of μTL complements the work of Lipeck, Saake, *et al.* [9, 11–15] on enforcement of temporal integrity constraints formulated using *biquantified* future temporal formulas[4]. The lower bound result in this paper shows that neither this approach, nor any other approach, can be extended to full FutureμTL while maintaining bounded size of the residual histories. In particular, it proves that there cannot be a finite automaton that can accept histories that satisfy a particular FutureμTL sentence[5].

Infinite histories and Potential Satisfaction. The paper only considers *finite* histories. However, there is an alternative to this approach: finite histories can be considered to be *finite prefixes* of infinite (or complete) histories. Queries are then evaluated with respect to the infinite histories (using essentially the

[4] A biquantified future TL formula allows quantification over the data domain either only outside the scope of temporal connectives or fully within all temporal connectives.

[5] Such an automaton always exists when μTL is restricted to the propositional setting [22, 23].

same semantic definitions; the only difference is that an infinite temporal domain is allowed for histories).

However, since only finite portion (a prefix) of the history is available at a particular (finite) point in time, we need to define answers to queries with respect to *possible completions* of the prefix to a complete history. Such approaches, in the first-order setting, run into serious decidability problems; Chomicki and Niwinski have shown that potential satisfaction is not decidable for closed PastTL formulas nor for biquantified formulas FutureTL formulas [3]. Thus considering query evaluation in this setting is not feasible for any sufficiently rich temporal query language (for more in-depth discussion see [21]).

4 An Expiration Operator for Past Fixpoint TL

In this section we develop an expiration operator for the PastμTL fragment and show that this operator guarantees a bounded history encoding for the residual history. The approach is based on an extension of the ideas of Chomicki [2, 4] for the past fragment of first-order temporal logic with the traditional **since** and $\bullet$ connectives.

The proposed operator maintains a sufficient amount of *temporal* information needed to evaluate a given PastμTL query as an instance of an *extended database schema.*

Definition 6 (Extended Database Schema)
Let φ be a PastμTL *query over a database schema ρ. We define*

$$\rho \cup \{r_\psi(x_1, \ldots, x_k) : \bullet\psi \text{ is a subformula of } \varphi \\ \text{with free variables } x_1, \ldots, x_k\}$$

to be the extended database schema *for φ. The predicate symbols r_ψ must not appear in ρ. We call predicates that belong to ρ* user-defined *and the new predicates r_ψ* auxiliary*.*

The expiration operator is defined as a set of *materialized view* maintenance rules. The definition of the rules is based on the *unfolding property* of the fixpoint operator, $\mu X.Q \equiv Q(\mu X.Q)$. Before we define the operator itself, we need an auxiliary technical definition.

Definition 7 *Let φ be a* PastμTL *formula of the form $\mu X.\psi$ such that ϑ is a subformula of ψ containing the fixpoint variable X free. We define the X-closure of ϑ in φ to be the formula $\vartheta[X/\mu X.\psi]$, i.e., in which all occurrences of X have been replaced by their fixpoint* definition*.*

For ψ a subformula of φ with free fixpoint variables $X_1, \ldots, X_k$ we define the closure (with respect to the fixpoint variables and φ) to be the PastμTL *formula $\psi[X_1/\mu X_1.\psi_1, \ldots, X_k/\mu X_k.\psi_k]$.*

This definition guarantees, that for every subformula ψ of a given PastμTL query φ, we can find a PastμTL formula ψ' in which all fixpoint variables X_i are in the scope of the appropriate fixpoint operator. Due to the unfolding rule, such a formula always exists and is equivalent to the original formula in the context of φ.

Definition 8 (Expiration Operator)
Let $\rho = (r_1, \ldots, r_k)$ be a database schema and Q a PastμTL *query. We define an expiration operator $\mathcal{E}$ for Q as follows:*

- *We define $\emptyset^{\mathcal{E}}$ to be a constant instance of the extended schema defined as*
$$\emptyset^{\mathcal{E}} = (\emptyset, \ldots, \emptyset, \emptyset, \ldots, \emptyset);$$
- *Let $D = (R_1, \ldots, R_k)$ be a new state of the history, and $\bullet\psi_1, \ldots, \bullet\psi_l$ be all temporal subformulas of Q. We define the $\Delta^{\mathcal{E}}$ operator as follows*
$$\begin{array}{ll} \Delta^{\mathcal{E}}(\mathcal{E}(H), D) = (R_1, \ldots, R_k, \emptyset, \ldots, \emptyset) & \text{for } H = \langle\,\rangle \\ \Delta^{\mathcal{E}}(\mathcal{E}(H), D) = (R_1, \ldots, R_k, \psi_1^{\mathcal{E}}(\mathcal{E}(H)), \ldots, \psi_l^{\mathcal{E}}(\mathcal{E}(H))) & \text{otherwise,} \end{array}$$
where $\psi_i^{\mathcal{E}}$ is the closure of ψ_i in which all occurrences of temporal subformulas $\bullet\vartheta$ were replaced by r_ϑ.
- *We define $Q^{\mathcal{E}}$ to be the query Q in which all occurrences of temporal subformulas $\bullet\vartheta$ were replaced by r_ϑ.*

Note that the definition of $\Delta^{\mathcal{E}}$ has to take into account the *special* treatment of the temporal connective $\bullet$ on state 0 (cf. Definition 3) by making explicitly the instances of the auxiliary predicate symbols r_ψ corresponding to $\bullet\psi$ empty.

Theorem 9 *$\mathcal{E}$ is an expiration operator for PastμTL that guarantees bounded encoding of residual histories. Moreover, for guarded* PastμTL *formulas, $Q^{\mathcal{E}}$ and the components of $\Delta^{\mathcal{E}}$ are first-order queries.*

Proof (sketch): Correctness, $Q(H) = Q^{\mathcal{E}}(\mathcal{E}(H))$, follows by induction on the length of H and the observation that the view maintenance rules $\psi_i^{\mathcal{E}}$ maintain the instance of r_ψ to be equivalent to $\bullet\psi$ in every state of H.

Bounded history encoding property follows from observing that both $Q^{\mathcal{E}}$ and all the re-materialization rules $\varphi_i^{\mathcal{E}}$ in the definition of $\Delta^{\mathcal{E}}$ only refer to the last instance of the extended database schema and to the current instance of the user-defined predicates—all of which can be represented by finite instances of the extended *relational* schema—we can easily show that the residual history size does not depend on the length of H.

For guarded formulas, all occurrences of the fixpoint variables are in the scope of $\bullet$ and thus are replaced by the auxiliary predicates. This guarantees $Q^{\mathcal{E}}$ and the rematerialization rules $\psi_i^{\mathcal{E}}$ that are used to define $\Delta^{\mathcal{E}}$ are first-order. □

Example 10 Applying the above construction on the fixpoint definition of the standard **since** temporal operator,
$$\mu X.\varphi \wedge \bullet(\psi \vee X),$$

that contains a single temporal subformula, $\bullet(\psi \vee X)$, proceeds as follows. We use an auxiliary predicate $r_{\psi\vee X}$ in the extended schema to maintain the "past" instance of $\bullet(\psi \vee X)$. The currency of the auxiliary relation after a history extension is achieved using a rule based on the closure of the subformula $\bullet(\psi\vee X)$ with respect to X,

$$\bullet(\psi \vee \mu X.\varphi \wedge \bullet(\psi \vee X)),$$

yielding

$$\begin{aligned} r_{\psi\vee X} &:= \psi \vee (\mu X.\varphi \wedge \bullet(\psi \vee X)) \\ &= \quad \psi \vee (\mu X.\varphi \wedge r_{\psi\vee X}) \quad = \psi \vee (\varphi \wedge r_{\psi\vee X}). \end{aligned}$$

This is essentially the same materialization rule as the rule designed by Chomicki specifically for the **since** connective [2] and implemented using active ECA rules in IBM Starburst by Chomicki and Toman [4].

PastμTL, however, allows queries that cannot be expressed in PastTL [24].

Example 11 Consider the query

$$\mu X.r(x) \vee \bullet\bullet X$$

that collects values in any instance of r for the even states of the history (counted from the end of the history). Note that to do this, we indeed need to collect the values for odd states too. This is automatically achieved by our construction. The temporal subformulas of this query are $\psi_1 = \bullet\bullet X$, and $\psi_2 = \bullet X$. The inductive definitions for auxiliary relations $r_{\bullet X}$ and r_X,

$$r_{\bullet X}(x) := r_X(x) \quad \text{and} \quad r_X(x) := r(x) \vee r_{\bullet X}(x),$$

capture exactly the required sets of values in even and odd states, respectively.

From an implementation point of view, we could use techniques based on ECA (active) database rules to maintain the instances of the auxiliary predicates up to date, similarly to the approach for enforcing integrity constraints formulated in PastTL proposed by Chomicki and Toman [4]. The only difference is in handling rematerialization rules that mutually depend on each other (cf. Example 11), where temporary storage is needed during (a sequential) execution of the *simultaneous assignment*. However, in the case of PastTL, the technique proposed in this paper does not need any such temporary storage, since all the rematerialization rules can be stratified with respect to the "subformula" relationship [4].

Note also, that the rematerialization rules never refer to the *new* state of the user-defined predicate symbols. Thus, it is sufficient to have only *one instance* of these predicates at any time. To achieve this, and unlike [4], the rematerialization rules must be executed *before* a new state is added to the history. This way, the instances of the auxiliary predicates are *advanced* to the next state before changes are made to the instances of the user-defined predicates. This way, the new instances of the auxiliary predicates can be used to answer the maintained query *during* the execution of an update transaction against the user-defined predicates (which will eventually advance these instances from the current state to the next state).

5 A Lower Bound for Future Fixpoint TL

The situation for FutureμTL is completely different since the *unfolding* rule for fixpoints extends toward the (unknown) future of the history. Indeed we show that this difference implies a *linear lower bound* on the size of the residual history.

Example 12 Consider the following FutureμTL query:

$$\varphi = \exists x, y.\Diamond(q(x,y) \wedge \mu X.r(x,y) \vee \bigcirc \exists z.X(x,z) \wedge X(z,y))$$

and all database histories H over the schema $\langle q(x,y), r(x,y)\rangle$ of the form

State	Database instance
0	$\{\}$ or $\{q(a,b)\}$
$\vdots$	$\vdots$
n	$\{\}$ or $\{q(a,b)\}$
$n+1$	$\{r(a,c_1),\ldots,r(c_i,c_{i+1}),\ldots,r(c_k,b)\}$,

where the constants a, b, and c_j are distinct values in $\mathbf{D}_H$. In other words, in each of the states $0,\ldots,n$, a single fact $q(a,b)$ is either true or false and the instances of r are empty, and in state $n+1$ the instance of r contains a chain of $k+1$ pairs linking the constants a and b. It is easy to see that the above query φ evaluates to true on a history of this form (i.e., $H, 0 \models \varphi$) if and only if

1. in state i the atomic fact $q(a,b)$ is true (for some $0 \leq i \leq n$),
2. there are distinct elements $c_1,\ldots,c_k$ in the domain $\mathbf{D}_H$ such that the atomic facts $r(a,c_1), r(c_1,c_2),\ldots,r(c_k,b)$ are true in state $n+1$, and
3. $k = 2^{n-i} - 1$.

Now let us consider the prefixes H' of all possible histories H of length n, i.e., missing the $(n+1)$-st (last) state. The size of the active domain of the prefixes H' is

$$|\mathbf{D}_{H'}| \leq |\{a,b\}| = 2,$$

and is independent of the length of this history, n. However, the above example shows that adding H's $(n+1)$-st state to H' allows us to "measure" how far in the history we moved past a state containing $q(a,b)$. Thus, to be able to preserve answers of φ under extensions of H' (in particular, to H), we have to retain at least one bit for each of the states $0,\ldots,n$ to record whether $q(a,b)$ was true or false in that state. Therefore, we need at least $\Omega(n) = \Omega(|\mathbf{T}_{H'}|)$ bits to represent the residual history. This observation yields the following theorem.

Theorem 13 *The worst-case space needed for residual histories produced by an arbitrary expiration operator for* FutureμTL *is bounded from below by a linear function in the length of the original history.*

An immediate consequence of this theorem is a lower bound for full μTL.

Corollary 14 *The space needed for residual histories produced by an arbitrary expiration operator for μTL is bounded from below by $\Omega(|\mathbf{T}_H|)$.*

Expiration Operators for Fragments of FutureμTL. On the other hand, several fragments of FutureμTL that admit bounded history encoding are already known:

- First order temporal logic (FOTL, with both past and future operators) can be embedded into the two-sorted first order logic, 2-FOL. This way, expiration techniques for 2-FOL [20] can be used to define an expiration operator guaranteeing bounded representation for residual histories.
- Biquantified future TL formulas can be handled using automata inspired approaches [9, 11–15].

The exact boundary between fragments of FutureμTL for which bounded expiration operators exist and those for which such operators cannot exist is not known at this time.

Note, however, that Theorem 13 precludes extending results on expiration of two sorted first-order logic (2-FOL) [20] to 2-FOL extended with general fixpoints while maintaining the existence of bounded history encoding.

Corollary 15 *The space needed for residual histories produced by an arbitrary expiration operator for a fixpoint extension of 2-FOL is bounded from below by $\Omega(|\mathbf{T}_H|)$.*

6 Conclusion

We have shown a significant difference between the behaviour of the past and the future fragments of μTL, when used as a query language for querying database histories. In particular, we have shown that large parts of a history, in an append-only setting, can be expired while maintaining answers to a finite set of queries formulated in the past fragment. On the other hand, for queries in the future fragment, one essentially needs to retain the whole history.

6.1 Future Work

While the picture for the past fragment is essentially settled, the results for FutureμTL are not quite satisfactory: there is a large gap between the biquantified future TL formulas (for which techniques proposed by Lipeck, Saake, *et al.* [9, 11–15] can be applied) and full FutureμTL for which there cannot be a bounded expiration operator.

One way in which we can limit the power of FutureμTL, in particular, is by limiting its ability to introduce (unbounded number of) new data variables. A natural restriction along these lines is to consider a fragment in which fixpoint operators are restricted to the temporal domain only. For example, ETL [24] is is an extension of LTL that allows temporal connectives (over discrete time domain) to be defined in terms of *regular expressions*. This extension allows expressing properties like "p is true in every even state", that are known to be inexpressible in LTL. FOETL is a first-order extension of ETL. Because the temporal connectives are defined *without* being able to use quantifiers over the data domain $\mathbf{D}_H$, this logic is essentially equivalent to the fragment of FutureμTL in which free fixpoint variables cannot appear in the scope of data quantifiers. We conjecture the following.

Conjecture 16 *There is an expiration operator for future FOETL that guarantees bounded history encoding of residual histories.*

Another way of limiting the power of μTL is to consider decidable fragments of first order, and possibly fixpoint, temporal logics [8]. These fragments may also allow bounded history encoding with respect to potential satisfaction (and thus potential answers to queries).

On a more practical side, we focus on the possibilities of combining nested fixpoint operators in PastμTL formulas, possibly yielding an expiration operator that does not require temporary storage for the past instance of the auxiliary predicates. It is an open question for how large fragment of PastμTL such an operator exists; it is easy to see that this can be achieved for various simple fragments of PastμTL, e.g., for PastTL.

Acknowledgments

The author gratefully acknowledges the Natural Sciences and Engineering Research Council of Canada, the Communications and Information Technology of Ontario and Nortel Networks Ltd. for their support of this research.

References

1. Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995.
2. Jan Chomicki. Efficient Checking of Temporal Integrity Constraints Using Bounded History Encoding. *ACM Transactions on Database Systems*, 20(2):149–186, 1995.
3. Jan Chomicki and Damian Niwinski. On the Feasibility of Checking Temporal Integrity Constraints. *Journal of Computer and System Sciences*, 51(3):523–535, 1995.
4. Jan Chomicki and David Toman. Implementing Temporal Integrity Constraints Using an Active DBMS. *IEEE Transactions on Data and Knowledge Engineering*, 7(4):566–582, 1995.
5. Jan Chomicki and David Toman. Temporal Logic in Information Systems. In Jan Chomicki and Gunter Saake, editors, *Logics for Databases and Information Systems*, pages 31–70. Kluwer, 1998.

6. Jan Chomicki, David Toman, and Michael H. Böhlen. Querying ATSQL databases with temporal logic. *ACM Transactions on Database Systems (TODS)*, 26(2):145–178, 2001.
7. Dov M. Gabbay, Ian M. Hodkinson, and Mark Reynolds. *Temporal Logic: Mathematical Foundations and Computational Aspects*. Oxford University Press, 1994.
8. Ian M. Hodkinson, Frank Wolter, and Michael Zakharyaschev. Decidable fragment of first-order temporal logics. *Annals of Pure and Applied Logic*, 106(1-3):85–134, 2000.
9. Klaus Hülsmann and Gunter Saake. Theoretical Foundations of Handling Large Substitution Sets in Temporal Integrity Monitoring. *Acta Informatica*, 28(4), 1991.
10. Christian S. Jensen. Vacuuming. In Richard T. Snodgrass, editor, *The TSQL2 Temporal Query Language*, pages 447–460. Kluwer Academic Publishers, 1995.
11. Udo W. Lipeck. Transformation of Dynamic Integrity Constraints into Transaction Specifications. *Theoretical Computer Science*, 76(1), 1990.
12. Udo W. Lipeck and Dasu Feng. Construction of Deterministic Transition Graphs from Dynamic Integrity Constraints. In *Graph-Theoretic Concepts in Computer Science*, pages 166–179. Springer-Verlag, LNCS 344, 1989.
13. Udo W. Lipeck, Michael Gertz, and Gunter Saake. Transitional Monitoring of Dynamic Integrity Constraints. *IEEE Data Engineering Bulletin*, June 1994.
14. Udo W. Lipeck and Gunter Saake. Monitoring Dynamic Integrity Constraints Based on Temporal Logic. *Information Systems*, 12(3):255–269, 1987.
15. Udo W. Lipeck and Heren Zhou. Monitoring Dynamic Integrity Constraints on Finite State Sequences and Existence Intervals. In *Workshop on Foundations of Models and Languages for Data and Objects. FMLDO'91*, pages 115–130, 1991.
16. Janne Skyt, Christian S. Jensen, and Leo Mark. A foundation for Vacuuming Temporal Databases . *Data and Knowledge Engineering*, 44(1):1–29, 2003.
17. Richard T. Snodgrass, editor. *The TSQL2 Temporal Query Language*. Kluwer, 1995.
18. David Toman. Point-based Temporal Extensions of SQL. In *International Conference on Deductive and Object-Oriented Databases*, pages 103–121, 1997.
19. David Toman. Point-Based Temporal Extensions of SQL and Their Efficient Implementation. In Opher Etzion, Sushil Jajodia, and Suryanarayana Sripada, editors, *Temporal Databases: Research and Practice*, pages 211–237. Springer LNCS State-of-the-Art Survey, 1998.
20. David Toman. Expiration of Historical Databases. In *International Symposium on Temporal Representation and Reasoning*, pages 128–135. IEEE Press, 2001.
21. David Toman. Logical Data Expiration. In Jan Chomicki, Gunter Saake, and Ron van der Meyden, editors, *Logics for Emerging Applications of Databases*, chapter 7. Springer, 2003.
22. Moshe Y. Vardi. A Temporal Fixpoint Calculus. In *ACM Symposium on Principles of Programming Languages*, pages 250–259, 1988.
23. Moshe Y. Vardi and Pierre Wolper. An Automata-Theoretic Approach to Automatic Program Verification. In *IEEE Symposium on Logic in Computer Science*, 1986.
24. Pierre Wolper. Temporal Logic Can Be More Expressive. *Information and Control*, 56(1/2):72–99, 1983.