Temporal Logic in Database Query Languages

Jan Chomicki, University at Buffalo, USA, `http://www.cse.buffalo.edu/~chomicki`
David Toman, University of Waterloo, Canada, `http://www.cs.uwaterloo.ca/~david`

DEFINITION

The term "temporal logic" is used, in the area of formal logic, to describe systems for representing and reasoning about propositions and predicates whose truth depends on time. These systems are developed around a set of *temporal connectives*, such as *sometime in the future* or *until*, that provide implicit references to time instants. First-order temporal logic is a variant of temporal logic that allows first-order predicate (relational) symbols, variables and quantifiers, in addition to temporal connectives. This logic can be used as a natural *temporal query language* for point-stamped temporal databases. A query (a temporal logic formula) is evaluated with respect to an *evaluation point (time instant)*. Each such point determines a specific database snapshot that can be viewed as a relational database. Thus, the evaluation of temporal logic queries resembles the evaluation of first-order (relational calculus) queries equipped with an additional capability to "move" the evaluation point using *temporal connectives*. In this way, it becomes possible to refer in a single query to multiple snapshots of a given temporal database. The answer to a temporal logic query evaluated with respect to all time instants forms a point-stamped temporal relation.

HISTORICAL BACKGROUND

Temporal Logic was developed, originally under the name of *tense logic*, by Arthur Prior in the late 1950s for representing and reasoning about natural languages. It was introduced to computer science by Amir Pnueli [8] as a tool for formal verification of software systems. The first proposal for using a temporal logic in the database context was by Sernadas [9]. Subsequently, de Castilho *et al.* [3] initiated the study of temporal-logic integrity constraints in relational databases. Tuzhilin and Clifford [12] considered temporal logic as a query language for temporal databases.

SCIENTIFIC FUNDAMENTALS

Temporal Logic is a variant of *modal* logic, tailored to expressing statements whose truth is relative to an underlying time domain which is commonly a *linearly ordered* set of time instants (see the entry Time Domain). The modalities are expressed using natural-language statements of the form *sometime in the future*, *always in the future*, etc., and are captured in the syntax of the logic using *temporal connectives*.

Temporal logics are usually rooted in *propositional logic*. However, for the purposes of querying (single-dimensional, valid time) point-stamped temporal databases, *Linear-Time First-Order Temporal Logic (FOTL)*, an extension of first-order logic (relational calculus) with *temporal connectives*, is used. More formally, given a relational schema ρ (of a snapshot of a point-stamped temporal database), the syntax of FOTL queries is defined as follows:

$$Q ::= r(x_{i_1},\ldots,x_{i_k}) \mid x_i = x_j \mid Q \wedge Q \mid \neg Q \mid \exists x.Q \mid Q \textbf{ since } Q \mid Q \textbf{ until } Q$$

for $r \in \rho$. The additional **since** and **until** connectives are the *temporal connectives*. The connectives completely *encapsulate* the structure of time: FOTL only refers to time *implicitly* using these connectives. In contrast, Temporal Relational Calculus (TRC) uses *explicit* temporal variables and attributes to refer to time. Note that there are no restrictions on the nesting of the temporal connectives, the Boolean connectives, and the quantifiers.

The meaning of all temporal logic formulas is defined relative to a particular time instant called the *evaluation point*. Intuitively, the evaluation point can be viewed as representing the *current instant* or *now*. The standard first-order parts of FOTL formulas are then evaluated in the snapshot of the temporal database determined by the evaluation point. The temporal connectives make it possible to change the evaluation point, i.e., to "move" it to the past or to the future. In this way the combination of first-order constructs and temporal connectives allows to ask queries that refer to multiple snapshots of the temporal database. For example the query Q_1 **since** Q_2 asks for all answers that make Q_2 true sometime in the past and Q_1 true between then and now. Similarly, queries about the future are formulated using the **until** connective. More formally, answers to FOTL queries are defined using a *satisfaction relation* that, for a given FOTL query, links a temporal database and an evaluation point (time instant) with the valuations that make the given query true with respect to the snapshot of that database at that particular evaluation point. This definition, given below, extends the standard definition of satisfaction for first-order logic.

Definition. [FOTL Semantics] Let DB be a point-stamped temporal database with a data domain D, a point-based time domain T_P, and a (snapshot) schema ρ.

The *satisfaction relation* $DB, \theta, t \models Q$, where Q is an FOTL formula, θ a valuation, and $t \in \mathrm{T}_P$, is defined inductively with respect to the structure of the formula Q:

$$\begin{array}{ll}
DB,\theta,t \models r_j(x_{i_1},\ldots,x_{i_k}) & \text{if } (\theta(x_{i_1}),\ldots,\theta(x_{i_k})) \in \mathbf{r}_j^{DB(t)} \\
DB,\theta,t \models x_i = x_j & \text{if } \theta(x_i) = \theta(x_j) \\
DB,\theta,t \models Q_1 \wedge Q_2 & \text{if } DB,\theta,t \models Q_1 \text{ and } DB,\theta,t \models Q_2 \\
DB,\theta,t \models \neg Q_1 & \text{if not } DB,\theta,t \models Q_1 \\
DB,\theta,t \models \exists x_i . Q_1 & \text{if there is } a \in D \text{ such that } DB,\theta[x_i \mapsto a],t \models Q_1 \\
DB,\theta,t \models Q_1 \ \mathbf{since}\ Q_2 & \text{if } \exists t_2 . t_2 < t \text{ and } DB,\theta,t_2 \models Q_2 \text{ and } (\forall t_1 . t_2 < t_1 < t \text{ implies } DB,\theta,t_1 \models Q_1) \\
DB,\theta,t \models Q_1 \ \mathbf{until}\ Q_2 & \text{if } \exists t_2 . t_2 > t \text{ and } DB,\theta,t_2 \models Q_2 \text{ and } (\forall t_1 . t_2 > t_1 > t \text{ implies } DB,\theta,t_1 \models Q_1)
\end{array}$$

where $\mathbf{r}_j^{DB(t)}$ is the instance of the relation r_j in the snapshot $DB(t)$ of the database DB at the instant t.

The answer to an FOTL query Q over DB is the set of tuples:

$$Q(DB) := \{(t, \theta(x_1), \ldots, \theta(x_k)) : DB, \theta, t \models Q\},$$

where $x_1, \ldots, x_k$ are the free variables of Q.

Note that answers to FOTL queries are (valid-time) point-stamped temporal relations.

Other commonly used temporal connectives, such as $\Diamond$ (*sometime in the future*), $\Box$ (*always in the future*), $\blacklozenge$ (*sometime in the past*), and $\blacksquare$ (*always in the past*) can be defined in terms of **since** and **until** as follows:

$$\begin{array}{rl}
\Diamond X_1 := \text{true } \mathbf{until}\ X_1 & \quad \blacklozenge X_1 := \text{true } \mathbf{since}\ X_1 \\
\Box X_1 := \neg \Diamond \neg X_1 & \quad \blacksquare X_1 := \neg \blacklozenge \neg X_1
\end{array}$$

For a discrete linear order, the $\bigcirc$ (*next*) and $\bullet$ (*previous*) operators are defined as follows:

$$\bigcirc X_1 := \text{false } \mathbf{until}\ X_1 \qquad \bullet X_1 := \text{false } \mathbf{since}\ X_1$$

The connectives **since**, $\blacklozenge$, $\blacksquare$, and $\bullet$ are called *past temporal connectives* (as they refer to the past) and **until**, $\Diamond$, $\Box$, and $\bigcirc$ *future temporal connectives*.

Example. The sensor information about who enters or exits a room is kept in the relations *Entry* (Figure 1a) and *Exit* (Figure 1b). Consider the query Q_a: *"For every time instant, who is in the room Bell 224 at that instant?"* It can be written in temporal logic as:

$$\exists r.\ ((\neg Exit(r,p))\ \mathbf{since}\ Entry(r,p)) \wedge r = ''\text{Bell 224}''$$

The answer to Q_a in the given database is presented in Figure 1c, under the assumption that time instants correspond to minutes.

Extensions

Several natural extensions of FOTL are obtained by modifying various components of the language:

a)

Entry		
Time	*Room*	*Person*
8 : 30	Bell 224	John
9 : 00	Bell 224	Mary
11 : 00	Capen 10	John

b)

Exit		
Time	*Room*	*Person*
10 : 30	Bell 224	John
10 : 00	Bell 224	Mary
12 : 00	Capen 10	John

c)

Time	*Person*
8 : 31	John
...	...
10 : 30	John
9 : 01	Mary
...	...
10 : 00	Mary

Figure 1: The *Entry* relation, the *Exit* relation, and the *Who is in Bell 224* relation.

Multiple Temporal Contexts (Multi-dimensional TLs). Standard temporal logics use a single *evaluation point* to relativize the truth of formulas with respect to time. However, there is no principled reason why not to use more than one evaluation point simultaneously. The logics taking this path are called *multidimensional temporal logics* or, more precisely, *n-dimensional* temporal logics (for $n \geq 1$). The satisfaction relation is extended, for a particular n, in a natural way, as follows:

$$DB, \theta, t_1, \ldots, t_n \models Q$$

where $t_1, \ldots, t_n$ are the evaluation points. In a similar fashion the definitions of *temporal connectives* are extended to this setting. Two-dimensional connectives, for example, seem to be the natural basis for temporal logic-style query language for the bitemporal data model. Unfortunately, there is no consensus on the selection of two-dimensional temporal operators.

An interesting variant of this extension are *interval temporal logics* that associate truth with *intervals*—these, however, can be considered *pairs* of time points [5, 13].

More Complex Time Domains. While most temporal logics assume that the time domain is equipped with a linear order only, in many practical settings the time domain has additional structure. For example, there may be a way to refer to *duration* (the distance of two time points). The linear-order temporal connectives are then generalized to *metric temporal connectives*:

$$\begin{array}{l} DB, \theta, t \models Q_1 \ \mathbf{since}_{\sim m}\ Q_2 \text{ if } \exists t_2.t - t_2 \sim m \text{ and } DB, \theta, t_2 \models Q_2 \text{ and } (\forall t_1.t_2 < t_1 < t \text{ implies } DB, \theta, t_1 \models Q_1) \\ DB, \theta, t \models Q_1 \ \mathbf{until}_{\sim m}\ Q_2 \text{ if } \exists t_2.t_2 - t \sim m \text{ and } DB, \theta, t_2 \models Q_2 \text{ and } (\forall t_1.t_2 > t_1 > t \text{ implies } DB, \theta, t_1 \models Q_1) \end{array}$$

for $\sim \in \{<, \leq, =, \geq, >\}$. Intuitively, these connectives provide means of placing constraints on how far in the past/future certain subformulas must be true. The resulting logic is then the *Metric First-order Temporal Logic*, a first-order variant of *Metric Temporal Logic* [7].

Example. To demonstrate the expressive power of Metric Temporal Logic, consider the query Q_b: *"For every time instant, who has been in the room Bell 224 at that instant for at least 2 hours?"*:

$$\exists r.\ ((\neg Exit(r,p))\ \mathbf{since}_{\geq 2:00}\ Entry(r,p)) \wedge r = ''\text{Bell 224}''$$

More Powerful Languages for Defining Temporal Connectives. Another extension introduces a more powerful *language* for specifying temporal connectives over the underlying linearly-ordered time domain. Vardi and Wolper show that temporal logics with connectives defined using first-order formulas cannot express various natural conditions such as "every other day". To remedy this shortcoming, they propose temporal connectives defined with the help of *regular expressions* (ETL [15]) or *fixpoints* (temporal μ-calculus [14]). Such extensions carry over straightforwardly to the first-order setting.

More Complex Underlying Query Languages. Last, instead of relational calculus, temporal connectives can be added to a more powerful language, such as Datalog. The resulting language is called *Templog* [2]. With suitable restrictions, query evaluation in this language is decidable and the language itself is equivalent to Datalog$_{1S}$ [2].

Expressive Power

The **since** and **until** temporal connectives can be equivalently defined using *formulas* in the underlying theory of linear order as follows:

$$\begin{aligned} X_1 \textbf{ since } X_2 &:= \exists t_2 . t_0 > t_2 \wedge X_2 \wedge \forall t_1 (t_0 > t_1 > t_2 \rightarrow X_1) \\ X_1 \textbf{ until } X_2 &:= \exists t_2 . t_0 < t_2 \wedge X_2 \wedge \forall t_1 (t_0 < t_1 < t_2 \rightarrow X_1) \end{aligned}$$

where X_1 and X_2 are *placeholders* that will be substituted with other formulas to be evaluated at the time instants t_1 and t_2, respectively. This observation indicates that every FOTL query can be equivalently expressed in TRC. The explicit translation parallels the inductive definition of FOTL satisfaction, uniformly parameterizing the formulas by t_0. In this way, an atomic formula $r(x_1, \ldots, x_k)$ (where r is a non-temporal snapshot relation) becomes $R(t_0, x_1, \ldots, x_k)$ (where R is a point-timestamped relation), and so on. For a particular t_0, evaluating $r(x_1, \ldots, x_k)$ in $DB(t_0)$, the snapshot of the database DB at t_0, yields exactly the same valuations as evaluating $R(t_0, x_1, \ldots, x_k)$ in DB. The embedding of the temporal connectives uses the definitions above. For example, the embedding of the **since** connective looks as follows:

$$\begin{aligned} \text{Embed}(Q_1 \textbf{ since } Q_2) \quad = \quad & \exists t_2 (t_0 > t_2 \wedge \forall t_0 (t_2 = t_0 \rightarrow \text{Embed}(Q_2)) \wedge \\ & \forall t_1 (t_0 > t_1 > t_2 \rightarrow \forall t_0 (t_1 = t_0 \rightarrow \text{Embed}(Q_1)))) \end{aligned}$$

Note that t_0 is the only free variable in Embed(Q_1), Embed(Q_2), and Embed(Q_1 **since** Q_2). Thus, in addition to applying the (first-order) definition of the temporal connective, the embedding performs an additional *renaming* of the temporal variable denoting the evaluation point for the subformulas, because the only free variable outside of the expanded temporal connectives must be called t_0.

Additional temporal connectives can be defined using the same approach, as formulas in the underlying theory of the temporal domain. However, for linear orders—the most common choice for such a theory—Kamp [6] has shown that all the connectives that are first-order definable in terms of linear order can be readily formulated using **since** and **until**.

In addition, it is easy to see on the basis of the above embedding that all FOTL queries (and their subqueries) define point-stamped temporal relations. This *closure property* makes FOTL amenable to specifying operators for temporal relational algebra(s) over the point-stamped temporal model. On the other hand, many, if not most, other temporal query languages, in particular various temporal extensions of SQL, are based on TRC and use temporal variables and attributes to explicitly access to timestamps. These languages do not share the above closure property. Surprisingly, and in contrast to the propositional setting, one can prove that query languages based on FOTL are strictly weaker than query languages based on TRC [1, 11]. The query

SNAPSHOT EQUALITY:
"are there two distinct time instants at which a unary relation R contains exactly the same values?"

cannot be expressed in FOTL. On the other hand, SNAPSHOT EQUALITY can be easily expressed in TRC as follows:

$$\exists t_1, t_2 . t_1 < t_2 \wedge \forall x . R(t_1, x) \iff R(t_2, x)$$

Intuitively, the subformula "$\forall x . R(t_1, x) \iff R(t_2, x)$" in the above query requires the simultaneous use of *two* distinct evaluation points t_1 and t_2 in the scope of a universal quantifier, which is not possible in FOTL. The result can be rephrased by saying that FOTL and other temporal query languages closed over the point-stamped temporal relations fail to achieve (the temporal variant of) Codd's completeness. In particular, there cannot be a temporal relational algebra over the (single-dimensional) point-stamped temporal model that can express all TRC queries.

In addition, this weakness is inherent to other languages with *implicit access* to timestamps, provided the underlying time domain is a linear order. In particular:

- adding a finite number of additional temporal connectives defined in the theory of linear order (including constants) is not sufficient for expressing SNAPSHOT EQUALITY [11];
- introducing *multidimensional temporal connectives* [4], while sufficient to express SNAPSHOT EQUALITY, is still not sufficient to express all TRC queries [10]. This also means that in the bitemporal model, the associated query languages cannot simultaneously preserve closure with respect to bitemporal relations and be expressively equivalent to TRC;

- using temporal connectives that are defined by fixpoints and/or regular expressions (see the earlier discussion) is also insufficient to express SNAPSHOT EQUALITY. Due to their non-first-order nature, the resulting query language(s) are incomparable, in terms of their expressive power, to TRC [1].

The only currently known way of achieving first-order completeness is based on using temporal connectives defined over a time domain whose structure allows the definition of pairing and projections (e.g., integer arithmetic). In this way temporal connectives can use pairing to simulate an unbounded number of variables and in turn the full TRC. However, such a solution is not very appealing, as the timestamps in the intermediate temporal relations do not represent time instants, but rather (encoded) tuples of such instants.

KEY APPLICATIONS

The main application area of FOTL is in the area of temporal integrity constraints. It is based on the observation that a sequence of relational database states resulting from updates may be viewed as a snapshot temporal database and constrained using Boolean FOTL formulas. Being able to refer to the past states of the database, temporal logic constraints generalize dynamic constraints. Temporal logic is also influential in the area of design and analysis of temporal query languages such as temporal relational algebras.

CROSS REFERENCE

bitemporal data model, Datalog, point-stamped data model, relational calculus, temporal integrity constraints, temporal query languages, temporal relational calculus, temporal algebras, time domain, time instant, TSQL2, valid time.

RECOMMENDED READING

[1] S. Abiteboul, L. Herr, and J. Van den Bussche. Temporal Versus First-Order Logic to Query Temporal Databases. In *ACM Symposium on Principles of Database Systems*, pages 49–57, 1996.

[2] M. Baudinet, J. Chomicki, and P. Wolper. Temporal Deductive Databases. In A. Tansel, J. Clifford, S. Gadia, S. Jajodia, A. Segev, and R. T. Snodgrass, editors, *Temporal Databases: Theory, Design, and Implementation*, chapter 13, pages 294–320. Benjamin/Cummings, 1993.

[3] J. M. V. de Castilho, M. A. Casanova, and A. L. Furtado. A Temporal Framework for Database Specifications. In *International Conference on Very Large Data Bases, VLDB'82*, pages 280–291, 1982.

[4] D. Gabbay, I. Hodkinson, and M. Reynolds. *Temporal Logic: Mathematical Foundations and Computational Aspects*. Oxford University Press, 1994.

[5] V. Goranko, A. Montanari, and G. Sciavicco. A Road Map of Interval Temporal Logics and Duration Calculi. *Journal of Applied Non-Classical Logics*, 14(1-2):9–54, 2004.

[6] J. Kamp. *Tense Logic and the Theory of Linear Order*. PhD thesis, University of California, Los Angeles, 1968.

[7] R. Koymans. Specifying Real-Time Properties with Metric Temporal Logic. *Real-Time Systems*, 2(4):255–299, 1990.

[8] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems*. Springer-Verlag, 1992.

[9] A. Sernadas. Temporal Aspects of Logical Procedure Definition. *Information Systems*, 5:167–187, 1980.

[10] D. Toman. On Incompleteness of Multi-dimensional First-order Temporal Logics. In *International Symposium on Temporal Representation and Reasoning and International Conference on Temporal Logic*, pages 99–106, 2003.

[11] D. Toman and D. Niwinski. First-Order Queries over Temporal Databases Inexpressible in Temporal Logic. In *International Conference on Extending Database Technology (EDBT'96)*, Avignon, France, 1996.

[12] A. Tuzhilin and J. Clifford. A Temporal Relational Algebra as a Basis for Temporal Relational Completeness. In D. McLeod, R. Sacks-Davis, and H.-J. Schek, editors, *International Conference on Very Large Data Bases, VLDB'90*, pages 13–23, 1990.

[13] J. van Benthem. *The Logic of Time*. D.Reidel, 2nd edition, 1991.

[14] M. Y. Vardi. A Temporal Fixpoint Calculus. In *ACM Symposium on Principles of Programming Languages*, 1988.

[15] P. Wolper. Temporal Logic Can Be More Expressive. *Information and Control*, 56:72–99, 1983.