A Timed Predicate Temporal Logic Sequent Calculus

Javier Enríquez Mendoza[1], Sam Speight[1], and Vincent Rahli[1]

[1]University of Birmingham, UK

Abstract

Our society strongly depends on critical information infrastructures such as autonomous vehicles, blockchain applications, IoT infrastructures, etc. Because of the complexity of these systems, guaranteeing that they operate in a correct and timely fashion is hard to achieve. In this abstract, we present a logical framework that allows reasoning about timing properties of such systems, and demonstrate its applicability through a series of examples.

Temporal Logic Temporal logic is used to describe and reason about the behavior of systems over time. A prominent example of such a logic is Timed Propositional Temporal Logic (TPTL) [4, 1, 2], an extension of Linear Temporal Logic (LTL) [6] with explicit clock variables that facilitates reasoning about time-bound properties. TPTL has notably been extended with past operators (TPTL + Past) in [5], which allows for reasoning about both past and future events, as well as quantifiers in [1]. Other "timed" temporal logics have been proposed such as Timed CTL [3, Sec.9.2], which most notably extends CTL with clocks and constraints on clocks, and for which model checking algorithms exist. We focus here on designing a calculus for a variant of TPTL + Past with quantifiers, which is fully formalised in Agda.

The syntax of TPTL is as follows, where $\mathbf{U}$ and $\bigcirc$ are the usual "until" and "next" LTL operators; p is an atomic proposition; x is a clock variable; $c \in \mathbb{Q}$; $\sim \in \{\leq, <, =, >, \geq\}$ allows comparing two time expressions; and the "freeze" formula $x \cdot \varphi$ binds the current time to x in φ:

$$\varphi \quad ::= \quad p \mid \varphi \wedge \varphi \mid \neg\varphi \mid \varphi \,\mathbf{U}\, \varphi \mid \bigcirc\varphi \mid x \sim c \mid x \cdot \varphi$$

Temporal operators such as "eventually" and "always" are defined as usual: $\Diamond\varphi := \mathtt{true}\,\mathbf{U}\,\varphi$ and $\Box\varphi := \neg\Diamond\neg\varphi$. Using the "freeze" operator, one can then capture that the formula φ eventually occurs by "time" c: $x \cdot \Diamond(x \leq c \wedge \varphi)$. To understand this, let us consider the semantics of a subset of TPTL, given in terms of a mapping $\rho = (\sigma, \tau)$ from $\mathbb{N}$ to states, where a state is the pair of a valuation on atoms and a timestamp, a $i \in \mathbb{N}$, and a variable valuation v:

$$\begin{array}{lcl} \rho, i, v \models \varphi \,\mathbf{U}\, \psi & \iff & \exists j > i.(\rho, j, v \models \psi) \wedge \forall k \in [i, j).(\rho, k, v \models \varphi) \\ \rho, i, v \models x \sim c & \iff & \tau_i - v(x) \sim c \\ \rho, i, v \models x \cdot \varphi & \iff & \rho, i, v[x \mapsto \tau_i] \models \varphi \end{array}$$

The semantics of $x \cdot \Diamond(x \leq c \wedge \varphi)$ w.r.t. ρ, i, v states that there exists a time τ_j at which φ is true, for some $j > i$ such that $\tau_j \leq \tau_i + c$.

Our Calculus We now present an extension of TPTL + Past, with quantifiers, a more general comparison operator, a "discrete" semantics, and a sequent calculus.

The syntax of this calculus is as follows (some operators are omitted for space reasons), where $\mathbf{S}$ and $\mathbf{Y}$ are the "since" and "yesterday" past counterparts of $\mathbf{U}$ and $\bigcirc$; the **Data** type is used to capture the information exchanged between agents; i ranges over a set of agents Agent; d ranges over a set of data Data; A ranges over a set of sets of agents Agents; an atom a is either an atomic proposition p, or a "send" atom $\mathtt{send}(i, d, A)$ stating that agent i sent the data d to the set of agents A, or a "receive" atom $\mathtt{recv}(i, d, j)$ stating that agent i received the data d from agent j, or an "internal event" atom $\mathtt{inter}(i, d)$ stating that d happened at agent i. It can be observed that the set of comparison operators only includes strict less-than

and equality, unlike TPTL's operators. This is because the remaining operators can be defined using these two along with conjunction and disjunction, thanks to the fact that our comparison operators are not dependent on the current time.

$$\begin{array}{rcl@{\qquad}rcl} \varphi & ::= & a \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \neg\varphi \mid \varphi\,\mathbf{U}\,\varphi \mid \bigcirc\varphi \mid \varphi\,\mathbf{S}\,\varphi \mid \mathbf{Y}\,\varphi \mid t \sim t \mid x \cdot \varphi \mid \forall u : T.\varphi \mid \exists u : T.\varphi & & & \\ T & ::= & \mathbf{Agent} \mid \mathbf{Agents} \mid \mathbf{Data} & \sim & ::= & < \mid = \\ t & ::= & x \mid \mathtt{0} \mid t \bullet t \mid \mathtt{s}(t) & a & ::= & p \mid \mathtt{send}(i,d,A) \mid \mathtt{recv}(i,d,j) \mid \mathtt{inter}(i,d) \end{array}$$

Let us now adapt and extend TPTL's semantics to our calculus. There, formulas are interpreted w.r.t.: (1) a timestamp t; (2) a function r from timestamps to states, called a run; (3) an interpretation function π, which given a state and an atom, captures whether the atom is true in that state; and (4) a variable valuation (we only present a subset for space reasons):

$$\begin{array}{lcl} \pi, r, t, v \models a & \iff & \pi(r(t))(a) \\ \pi, r, t, v \models \varphi\,\mathbf{S}\,\psi & \iff & \exists t' < t.(\pi, r, t', v \models \psi) \wedge \forall t'' \in (t', t].(\pi, r, t'', v \models \varphi) \\ \pi, r, t, v \models \mathbf{Y}\,\varphi & \iff & t > 0 \text{ and } \pi, r, t-1, v \models \varphi \\ \pi, r, t, v \models t_1 \sim t_2 & \iff & [\![t_1]\!]_v \sim [\![t_2]\!]_v \\ \pi, r, t, v \models x \cdot \varphi & \iff & \pi, r, t, v[x \mapsto t] \models \varphi \\ \pi, r, t, v \models \forall u : T.\varphi & \iff & \pi, r, t, v[u \mapsto z] \models \varphi \text{ for all } z \in [\![T]\!] \end{array}$$

$$\begin{array}{lcl} [\![x]\!]_v & := & v(x) \\ [\![\mathtt{0}]\!]_v & := & 0 \\ [\![t_1 \bullet t_2]\!]_v & := & [\![t_1]\!]_v + [\![t_2]\!]_v \\ [\![\mathtt{s}(t)]\!]_v & := & [\![t]\!]_v + 1 \end{array} \qquad \begin{array}{lcl} [\![\mathbf{Agent}]\!] & := & \mathsf{Agent} \\ [\![\mathbf{Agents}]\!] & := & \mathsf{Agents} \\ [\![\mathbf{Data}]\!] & := & \mathsf{Data} \end{array}$$

Thanks to the more general comparison operator, a time bounded version of the $\Diamond$ operator can now be defined in a more straightforward way: $\Diamond_t\varphi := x \cdot \Diamond(y \cdot y \leq x \bullet t \wedge \varphi)$, stating that φ happens sometime in the future by "time" t (where $t_1 \leq t_2$ stands for $t_1 < t_2 \vee t_1 = t_2$).

Several proof systems have been proposed for LTL-like logics. However, the until operator is known to pose difficulties. A labeled Natural Deduction system was proposed in [7] to circumvent those difficulties. Simplifying and generalizing that system, we propose a labeled sequent calculus that allows reasoning about until and timing properties using the same machinery. We show below the until rules, which illustrate key aspects of our calculus:

$$\frac{\Gamma \vdash_{\mathtt{t}} \mathtt{t} < \mathtt{t}_1 \qquad \Gamma \vdash_{\mathtt{t}_1} \psi \qquad \Gamma, \mathtt{t} \leq x, x < \mathtt{t}_1 \vdash_x \varphi}{\Gamma \vdash_{\mathtt{t}} \varphi\,\mathbf{U}\,\psi}\ \mathbf{U}\,\mathrm{R} \qquad \frac{\Gamma, \mathtt{t} < x, (\psi)^x, (\varphi)^{[\mathtt{t},x)} \vdash_{t_1} \gamma}{\Gamma, (\varphi\,\mathbf{U}\,\psi)^{\mathtt{t}} \vdash_{t_1} \gamma}\ \mathbf{U}\,\mathrm{L}$$

A sequent is of the form $\Gamma \vdash_{\mathtt{t}} \varphi$, stating that φ holds at time $\mathtt{t}$ in the context Γ. An hypothesis is of the form $(\psi)^r$, where r is a time annotation, which can either be: (1) a time expression $\mathtt{t}$, (2) a time interval such as $[t_1, t_2]$ or $[t_1, t_2)$, or (3) an "empty" annotation stating that the hypothesis holds at all times. The $\mathbf{U}\,\mathrm{R}$ rule states that for $\varphi\,\mathbf{U}\,\psi$ to be true at time t it must be that there exists a time t_1 greater than t at which ψ holds, and that φ holds between t and t_1. The $\mathbf{U}\,\mathrm{L}$ rule states that if $\varphi\,\mathbf{U}\,\psi$ is true at time t then there must be a time x (a variable) greater than t at which φ holds, and φ must be true between t and x.

Let us illustrate this calculus through a straightforward example where all nodes are correct and communication is synchronous, ensuring that all sent messages are received within a time interval T, captured by the following formula (if an agent sends a message d at time t to a group, all members of that group will receive the message by time $t + T$):

$$\forall a : \mathbf{Agent}.\forall d : \mathbf{Data}.\forall A : \mathbf{Agents}.\forall b : \mathbf{Agent}.\Box(\mathtt{send}(a,d,A) \rightarrow \Diamond_T(b \in A \rightarrow \mathtt{recv}(a,d,b)))$$

Additionally, we assume that any information received by a node is immediately shared with other node (whenever an agent b receives a message, it immediately sends it to the set of nodes containing only c):

$$\forall a : \mathbf{Agent}.\forall d : \mathbf{Data}.\forall b : \mathbf{Agent}.\forall c : \mathbf{Agent}.\Box(\texttt{recv}(a, d, b) \rightarrow \texttt{send}(b, d, \{c\}))$$

Using our calculus, from the above assumptions, one can then derive the following formula, stating that if a node a sends a message d to node b, then c will also receive d before $2T$:

$$\forall a : \mathbf{Agent}.\forall d : \mathbf{Data}.\forall b : \mathbf{Agent}.\forall c : \mathbf{Agent}.\texttt{send}(a, d, \{b\}) \rightarrow \Diamond_{2T}(\texttt{recv}(b, d, c))$$

Based on this calculus, we now plan on designing a type system tailored to the unique demands of distributed systems, aligning type-based constraints with the above logical framework.

References

[1] Rajeev Alur and Thomas A. Henzinger. Real-time logics: complexity and expressiveness. In *LICS*, pages 390–401, June 1990. URL: https://ieeexplore.ieee.org/document/113764/?arnumber=113764, doi:10.1109/LICS.1990.113764.

[2] Rajeev Alur and Thomas A. Henzinger. A really temporal logic. *Journal of the ACM*, 41(1):181–203, January 1994. doi:10.1145/174644.174651.

[3] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, 2008.

[4] Patricia Bouyer, Fabrice Chevalier, and Nicolas Markey. On the expressiveness of tptl and mtl. In *Proceedings of the 25th International Conference on Foundations of Software Technology and Theoretical Computer Science*, FSTTCS '05, page 432–443, Berlin, Heidelberg, 2005. Springer-Verlag. doi:10.1007/11590156_35.

[5] Laura Bozzelli, Aniello Murano, and Loredana Sorrentino. Alternating-time temporal logics with linear past. *Theoretical Computer Science*, 813:199–217, April 2020. URL: https://linkinghub.elsevier.com/retrieve/pii/S0304397519307546, doi:10.1016/j.tcs.2019.11.028.

[6] Amir Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, pages 46–57, 1977. doi:10.1109/SFCS.1977.32.

[7] Marco Volpe. *Labeled natural deduction for temporal logics*. PhD thesis, University of Verona, Italy, 2010. URL: https://opac.bncf.firenze.sbn.it/bncf-prod/resource?uri=TD11087417.