Clock Specifications for Temporal Tasks in Planning and Learning

Giuseppe De Giacomo[1,2], Marco Favorito[3] and Fabio Patrizi[2]

[1]University of Oxford, UK
[2]Sapienza University of Rome, Italy
[3]Banca d'Italia, Italy

Abstract

Recently, Linear Temporal Logics on finite traces, such as LTL$_f$ (or LDL$_f$), have been advocated as high-level formalisms to express dynamic properties, such as goals in planning domains or rewards in Reinforcement Learning (RL). This paper addresses the challenge of separating high-level temporal specifications from the low-level details of the underlying environment (domain or MDP), by allowing for expressing the specifications at a different time granularity than the environment. We study the notion of a clock which progresses the high-level LTL$_f$ specification, whose ticks are triggered by dynamic (low-level) properties defined on the underlying environment. The obtained separation enables terse high-level specifications while allowing for very expressive forms of clock expressed as general LTL$_f$ properties over low-level features, such as counting or occurrence/alternation of special events. We devise an automata-based construction to compile away the clock into a deterministic automaton that is polynomial in the size of the automata characterizing the high-level and clock specifications. We show the correctness of the approach and discuss its application in several contexts, including FOND planning, RL with LTL$_f$ Restraining Bolts, and Reward Machines.

Keywords

Temporal Logics, Automata Theory, Planning and Learning for Temporal Tasks

1. Introduction

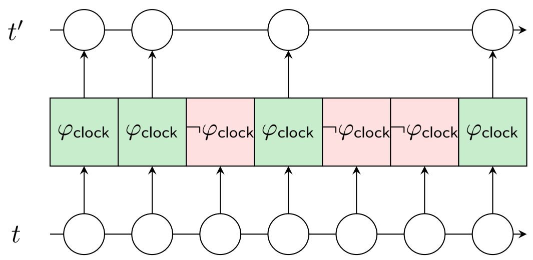
Linear Temporal Logic on finite traces (LTL$_f$) [1] has been advocated as a proper variant of LTL interpreted over finite traces. Moreover, at no cost of computational complexity but higher expressive power, the authors propose a novel formalism, Linear Dynamic Logic on finite traces (LDL$_f$); it is as expressive as regular expressions, while retaining the declarative nature and intuitive appeal of LTL$_f$. Both LTL$_f$ and LDL$_f$ have been quite successful in the AI and Formal Methods communities in recent years. For example, they have been used for finite temporal synthesis [2, 3, 4, 5], in Fully-Observable Non-Deterministic (FOND) Planning for LTL$_f$ Goals [6, 7, 8, 9], for reward function specification in the theory of Markov Decision Processes (MDP) [10, 11] and in Reinforcement Learning (RL) [12] with temporal logic rewards [13, 14].

OVERLAY 2023: 5th Workshop on Artificial Intelligence and Formal Verification, Logic, Automata, and Synthesis, November 7, 2023, Rome, Italy
giuseppe.degiacomo@cs.ox.ac.uk (G. D. Giacomo); marco.favorito@bancaditalia.it (M. Favorito); patrizi@diag.uniroma1.it (F. Patrizi)
0000-0001-9680-7658 (G. D. Giacomo); 0000-0001-9566-3576 (M. Favorito); 0000-0002-9116-251X (F. Patrizi)

The use of task specification languages, e.g., in the form of $\text{LTL}_f/\text{LDL}_f$ formulas, allowed greater richness in goal specifications, and improved modularity of the AI system by providing a clear separation between the goal and the environment. However, despite their successes, there is a crucial issue that, to the best of our knowledge, has not been studied yet. So far, it has been implicitly assumed that the time granularity of the task specification and the time granularity of the acting of the agent in the world are *synchronized.* In other words, each agent timestep is in one-to-one correspondence with each task timestep. While this assumption is not limiting in terms of *what* specifications can be expressed, we argue that it is limiting in terms of *how.* Conceptually, the synchrony assumption between the designer and the agent is not realistic, as these are two different entities which might have different cognitive systems, and therefore different perceptions of the world. In particular, the designer and the agent might have different *temporal processing capacities.* The task desired by the designer is expressed from the designer's perspective but has to be executed by the agent, which has its own understanding of the world and the task.

Consider the following scenario: a RL agent (a computer program) playing the Atari game Breakout [15], and a human designer that assigns the task of breaking the columns of bricks from left to right (as in [14]). The designer task can be expressed in LTL_f in terms of "next" operator, denoted with "$\circ$": $\circ(c_1 \wedge \circ(c_2 \wedge \dots))$. However, these two entities have completely different perceptions of the world. On the one hand, the RL agent observes the pixels of the game screen, and has access to the Atari Breakout simulator; hence, the timestep of the environment is under control of the agent itself. On the other hand, the designer has a common-sense understanding of the environment, and has proposed a task based on its perception. In particular, here we focus on the notion of what is the "next timestep" for such entities. While for the agent, the "next timestep" coincides with the "next frame", for the designer it makes more sense to consider more abstract or higher-level timesteps, such as "the next removed brick", or "the next removed column". Given this unavoidable discrepancy, the designer should instruct the agent about how to interpret the designer's task according to the time resolution of the agent's perception. Without any further instruction, the original designer's task cannot be correctly interpreted by the agent, because the meaning of the "next" operator is based on the agent's timestep resolution, i.e. the next frame. Therefore, the designer is forced to express the goal specification in terms of *stutter-invariant* operators [16], e.g. in terms of eventually operators: $\diamond(c_1 \wedge \diamond(c_2 \wedge \dots))$. The task specification might be more naturally expressed in a different time granularity than the agent's, but there must still be a sort of "glue" between the two granularities.

Related works. The topic of different temporal abstractions within the same information system has been investigated for decades in computer science. Several different formalisms to finitely represent infinite-time granularities have been proposed in the literature, based on algebraic [17, 18, 19, 20], logical [21, 22, 23, 24], string- based [25], and automaton-based [26, 27, 28, 29] approaches; see [30] for a survey on the topic. However, instead of devising ad-hoc temporal goal specification languages, or specific automata-based techniques, as the references above, we would like to keep intact both the LTL_f formalism and rely on classic automata theory, while allowing the designer to specify the clock specification and automated techniques to use it. This would give us broader impact in the wide community that is using LTL_f, and better reliance on the wide availability of supporting tools. Another line of research aimed to extend temporal logic with the so-called *clock operator* is described in [31, 32]. The



t'
φ_{clock}
φ_{clock}
$\neg\varphi_{\mathsf{clock}}$
φ_{clock}
$\neg\varphi_{\mathsf{clock}}$
$\neg\varphi_{\mathsf{clock}}$
φ_{clock}
t

Figure 1: How clock specifications work.

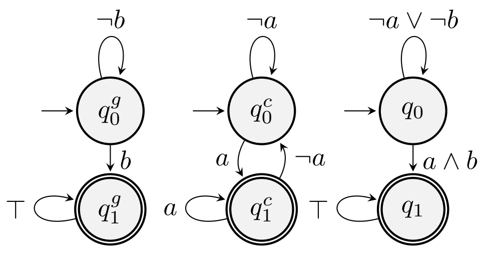


Figure 2: An example of clock product. From left to right: DFA for $\varphi_{\mathsf{goal}} = \Diamond b$, DFA for $\varphi_{\mathsf{clock}} = \Diamond(a \wedge \mathit{last})$, and minimized $\mathcal{A}_{\mathsf{goal}\times\mathsf{clock}}$.

clock operator was proposed in the context of modern hardware design, in which there is no notion of a single clock. Such an operator allows us to disambiguate which clock to use in order to evaluate a temporal formula or, in other words, what is the "next timestep". Both LTL$^{@}$ [31] and PSL [32] extend LTL to support clock operators. Again, our purpose is not to change the amenable syntax of LTL$_f$, but to provide a tool for AI designers to specify the timestep granularity for semantic evaluation. Moreover, in their case, the clock only depends on the current instant, while we consider the clock specified using temporal logic formulas.
Contributions. In this work, we are interested in the notion of *clock specification*, i.e. the explicit specification of what is "the next step" for the task given by the designer. The core contribution of this paper is to formalize and study the properties and expressivity of clock specifications in the context of temporal goal specifications. We formalize our approach by introducing a clock specification formula for a temporal goal, we show how these two can be compiled together in order to change the time granularity for the evaluation of the goal formula, by means of an automata-based construction. This technique can be used to solve the problem of temporal goal satisfaction, both in planning and in learning, in the presence of clocks.

2. Clock Specifications

Let $\mathcal{P}$ be a set of propositions that capture facets of interest. In the context of clock specifications, we have a LTL$_f$/LDL$_f$ formula φ_{goal} specifying the desired temporal task, i.e. the *goal formula*. In addition, we have a LTL$_f$/LDL$_f$ formula φ_{clock}, called *clock formula*, describing the timesteps to consider when evaluating the goal formula. We call the pair $(\varphi_{\mathsf{goal}}, \varphi_{\mathsf{clock}})$ *clocked specification* and say that φ_{goal} is *under clock specification* φ_{clock}. We assume, without loss of generality, that both φ_{clock} and φ_{goal} are defined over $\mathcal{P}$. Figure 1 intuitively explains the scenario we are considering. Circles represent trace timesteps. The bottom trace has finest time granularity t. The formula φ_{clock} is evaluated on every prefix of the trace. If the trace prefix at some time t_i makes the formula φ_{clock} true, then the timestep is passed to the evaluation of φ_{goal}, and becomes a timestep of the coarser-grained timestep sequence t'. On the other hand, if for some timestep t_i, the trace prefix up to that timestep does not satisfy φ_{clock}, then the current timestep is ignored at the higher level t'.

We now formalize the semantics of the evaluation of φ_{goal} under clock formula φ_{clock}. We start with the notion of trace *projection*. The *projection of* π *onto clock formula* φ_{clock} is the trace

$\pi|_{\varphi_{\mathsf{clock}}} = p_0, p_1, \ldots, p_n$, where $p_i = \pi[i]$, if $\pi(0, i+1) \models \varphi_{\mathsf{clock}}$, and $p_i = \epsilon$, otherwise. We define the *clocked semantics* of a LTL$_f$/LDL$_f$ formula φ under clock formula φ_{clock} in terms of the original semantics but considering projection of a trace π onto clock formula φ_{clock}. That is, we say that π *models* φ *under clock formula* φ_{clock}, written $\pi \models^{\varphi_{\mathsf{clock}}} \varphi$, iff $\pi|_{\varphi_{\mathsf{clock}}} \models \varphi$.

Now we introduce an automata-based construction to reason over clocked LTL$_f$/LDL$_f$ specifications. This technique will be useful for automata-based construction in planning and learning for LTL$_f$/LDL$_f$ goals. Let $(\varphi_{\mathsf{goal}}, \varphi_{\mathsf{clock}})$ be a LTL$_f$/LDL$_f$ clocked specification. Firstly, we compute the DFAs $\mathcal{A}_{\mathsf{goal}} = \langle Q_g, 2^{\mathcal{P}}, q_0^g, \delta_g, F_g \rangle$ and $\mathcal{A}_{\mathsf{clock}} = \langle Q_c, 2^{\mathcal{P}}, q_0^c, \delta_c, F_c \rangle$ of φ_{goal} and φ_{clock}, respectively. Then, we compute the *clocked product* $\mathcal{A}_{\mathsf{goal}\times\mathsf{clock}} = \langle Q', 2^{\mathcal{P}}, q_0', \delta', F' \rangle$, defined as follows: $Q' = Q_g \times Q_c$, $q_0' = (q_0^g, q_0^c)$, $F' = F_g \times Q_c$, $\delta'((q^g, q^c), a) = (\delta_g(q^g, a), \delta_c(q^c, a))$ if $\delta_c(q^c, a) \in F_c$, otherwise $(q^g, \delta_c(q^c, a))$. Intuitively, the clocked product is like the classical synchronous product between DFAs, except that the state component coming from the goal automaton q^g is progressed only if the clock component q^c transitions into an accepting state of $\mathcal{A}_{\mathsf{clock}}$. An example is shown in Figure 2. We have the following result:

Theorem 1. *Let* $(\varphi_{\mathsf{goal}}, \varphi_{\mathsf{clock}})$ *be a clocked specification, and* $\mathcal{A}_{\mathsf{goal}\times\mathsf{clock}}$ *be clocked product of* $\mathcal{A}_{\mathsf{goal}}$ *and* $\mathcal{A}_{\mathsf{clock}}$*. For any finite trace* π*,* $\pi \models^{\varphi_{\mathsf{clock}}} \varphi_{\mathsf{goal}}$ *iff* $\pi \in \mathcal{L}(\mathcal{A}_{\mathsf{goal}\times\mathsf{clock}})$

Theorem 1 tells us that clocked LDL$_f$ specifications are not more expressive than regular expressions and, therefore, than LDL$_f$. On the other hand, it is easy to see that LDL$_f$ is not more expressive than clocked LDL$_f$ specifications:

Theorem 2. *Given a* LTL$_f$/LDL$_f$ *formula* φ*, the clocked specification* (φ, tt) *is equivalent.*

We say that a formula ψ is *unclocked-equivalent* to φ under clock formula φ_{clock} if, for every trace π, we have $\pi \models^{\varphi_{\mathsf{clock}}} \varphi$ iff $\pi \models \psi$. Here we show that we can automatically find "unclocked" LTL$_f$/LDL$_f$ formulas that are semantically equivalent to clocked LTL$_f$/LDL$_f$ specifications.

Theorem 3. *Given a clocked specification* $(\varphi_{\mathsf{goal}}, \varphi_{\mathsf{clock}})$*, there exists a* LDL$_f$ *formula* ψ *that is unclocked-equivalent to* $(\varphi_{\mathsf{goal}}, \varphi_{\mathsf{clock}})$*.*

Proof sketch. Compute the regular expression r equivalent to $\mathcal{A}_{\mathsf{goal}\times\mathsf{clock}}$, and take $\psi = \langle r \rangle end$. Correctness follows by construction and by Theorem 1.

3. Discussion

We have sketched the theoretical bases for clock specifications for temporal tasks. This framework can be applied to FOND planning for LTL$_f$/LDL$_f$ goals [6], by using $\mathcal{A}_{\mathsf{goal}\times\mathsf{clock}}$ (instead of $\mathcal{A}_{\mathsf{goal}}$) in the cross-product with the DFA of the domain, or for specifying non-Markovian "clocked" rewards in Non-Markovian Reward Decision Processes (NMRDP) [10], by means of the usual product construction between the MDP and the reward specification represented by $\mathcal{A}_{\mathsf{goal}\times\mathsf{clock}}$. The same approach can be combined with logic-based reward specifications in a Reinforcement Learning setting, as in RL with Restraining Bolts [14, 33]; the reward is given only when both the goal formula and the clock formula are satisfied. A similar construction can be obtained when dealing with Reward Machines [34].

Acknowledgements

This work has been partially supported by the EU H2020 project AIPlan4EU (No. 101016442), the ERC-ADG White- Mech (No. 834228), the EU ICT-48 2020 project TAILOR (No. 952215), the PRIN project RIPER (No. 20203FFYLK), and the PNRR MUR project FAIR (No. PE0000013).

References

[1] G. De Giacomo, M. Y. Vardi, Linear temporal logic and linear dynamic logic on finite traces, in: IJCAI, IJCAI/AAAI, 2013, pp. 854–860.

[2] G. De Giacomo, M. Y. Vardi, Synthesis for LTL and LDL on finite traces, in: IJCAI, AAAI Press, 2015, pp. 1558–1564.

[3] G. De Giacomo, M. Y. Vardi, Ltl$_f$ and ldl$_f$ synthesis under partial observability, in: IJCAI, IJCAI/AAAI Press, 2016, pp. 1044–1050.

[4] A. Camacho, J. A. Baier, C. J. Muise, S. A. McIlraith, Finite LTL synthesis as planning, in: ICAPS, AAAI Press, 2018, pp. 29–38.

[5] S. Zhu, L. M. Tabajara, J. Li, G. Pu, M. Y. Vardi, Symbolic ltlf synthesis, in: IJCAI, 2017.

[6] G. De Giacomo, S. Rubin, Automata-theoretic foundations of FOND planning for ltlf and ldlf goals, in: IJCAI, ijcai.org, 2018, pp. 4729–4735.

[7] R. I. Brafman, G. De Giacomo, Planning for ltlf /ldlf goals in non-markovian fully observable nondeterministic domains, in: IJCAI, ijcai.org, 2019, pp. 1602–1608.

[8] A. Camacho, S. A. McIlraith, Strong fully observable non-deterministic planning with LTL and ltlf goals, in: IJCAI, ijcai.org, 2019, pp. 5523–5531.

[9] G. De Giacomo, M. Favorito, F. Fuggitti, Planning for temporally extended goals in pure-past linear temporal logic: A polynomial reduction to standard planning, CoRR abs/2204.09960 (2022).

[10] R. I. Brafman, G. De Giacomo, F. Patrizi, Ltlf/ldlf non-markovian rewards, in: AAAI, AAAI Press, 2018, pp. 1771–1778.

[11] R. I. Brafman, G. D. Giacomo, Regular decision processes: A model for non-markovian domains, in: IJCAI, ijcai.org, 2019, pp. 5516–5522.

[12] R. S. Sutton, A. G. Barto, Reinforcement learning - an introduction, Adaptive computation and machine learning, MIT Press, 1998.

[13] A. Camacho, R. T. Icarte, T. Q. Klassen, R. A. Valenzano, S. A. McIlraith, LTL and beyond: Formal languages for reward function specification in reinforcement learning, in: IJCAI, ijcai.org, 2019, pp. 6065–6073.

[14] G. De Giacomo, L. Iocchi, M. Favorito, F. Patrizi, Foundations for restraining bolts: Reinforcement learning with ltlf/ldlf restraining specifications, in: ICAPS, AAAI Press, 2019, pp. 128–136.

[15] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al., Human-level control through deep reinforcement learning, nature 518 (2015) 529–533.

[16] L. Lamport, What good is temporal logic?, in: IFIP Congress, North-Holland/IFIP, 1983, pp. 657–668.

[17] C. Bettini, S. Jajodia, X. S. Wang, Time granularities in databases, data mining, and temporal reasoning, Springer, 2000.

[18] B. Leban, D. McDonald, D. Forster, A representation for collections of temporal intervals, in: AAAI, Morgan Kaufmann, 1986, pp. 367–371.

[19] M. Niezette, J. Stevenne, An efficient symbolic representation of periodic time, in: Proceedings of the International Conference on Information and Knowledge Management (CIKM), 1992, pp. 161–168.

[20] P. Ning, X. S. Wang, S. Jajodia, An algebraic representation of calendars, Ann. Math. Artif. Intell. 36 (2002) 5–38.

[21] C. Combi, M. Franceschet, A. Peron, Representing and reasoning about temporal granularities, J. Log. Comput. 14 (2004) 51–77.

[22] S. Demri, LTL over integer periodicity constraints, Theor. Comput. Sci. 360 (2006) 96–123.

[23] H. Bowman, S. J. Thompson, A decision procedure and complete axiomatization of finite interval temporal logic with projection, J. Log. Comput. 13 (2003) 195–239.

[24] G. Hariharan, B. Kempa, T. Wongpiromsarn, P. H. Jones, K. Y. Rozier, MLTL multi-type (MLTLM): A logic for reasoning about signals of different types, in: NSV/FoMLAS@CAV, volume 13466 of *Lecture Notes in Computer Science*, Springer, 2022, pp. 187–204.

[25] J. Wijsen, A string-based model for infinite granularities, in: Proceedings of the AAAI Workshop on Spatial and Temporal Granularities, 2000, pp. 9–16.

[26] U. D. Lago, A. Montanari, Calendars, time granularities, and automata, in: SSTD, volume 2121 of *Lecture Notes in Computer Science*, Springer, 2001, pp. 279–298.

[27] D. Bresolin, A. Montanari, G. Puppis, Time granularities and ultimately periodic automata, in: JELIA, volume 3229 of *Lecture Notes in Computer Science*, Springer, 2004, pp. 513–525.

[28] U. D. Lago, A. Montanari, G. Puppis, Compact and tractable automaton-based representations of time granularities, Theor. Comput. Sci. 373 (2007) 115–141.

[29] U. D. Lago, A. Montanari, G. Puppis, On the equivalence of automaton-based representations of time granularities, in: TIME, IEEE Computer Society, 2007, pp. 82–93.

[30] J. Euzenat, A. Montanari, Time granularity, in: Handbook of Temporal Reasoning in Artificial Intelligence, volume 1 of *Foundations of Artificial Intelligence*, Elsevier, 2005, pp. 59–118.

[31] C. Eisner, D. Fisman, J. Havlicek, A. McIsaac, D. V. Campenhout, The definition of a temporal clock operator, in: ICALP, volume 2719 of *Lecture Notes in Computer Science*, Springer, 2003, pp. 857–870.

[32] C. Eisner, D. Fisman, A Practical Introduction to PSL, Series on Integrated Circuits and Systems, Springer, 2006.

[33] G. De Giacomo, M. Favorito, L. Iocchi, F. Patrizi, A. Ronca, Temporal logic monitoring rewards via transducers, in: KR, 2020, pp. 860–870.

[34] R. T. Icarte, T. Q. Klassen, R. A. Valenzano, S. A. McIlraith, Teaching multiple tasks to an RL agent using LTL, in: AAMAS, International Foundation for Autonomous Agents and Multiagent Systems Richland, SC, USA / ACM, 2018, pp. 452–461.