# An Introduction to First-Order Linear Temporal Logic

**Nicola Gigante**
Free University of Bozen-Bolzano, Italy

## Abstract

Linear temporal logic (LTL), most commonly defined as a propositional modal logic, is the de-facto standard language for specifying temporal properties of systems in formal verification, artificial intelligence, and other fields. First-order linear temporal logic (FOLTL) lifts LTL to the setting of first-order logic, obtaining a remarkably flexible and expressive formalism. First-order modal and temporal logics have a long history, but recent years have seen a rise of interest in (well-behaved fragments of) FOLTL for the specification of complex infinite-state systems. This tutorial is a gentle introduction to the field of first-order temporal logics, starting from classic results and exploring recent directions.

**2012 ACM Subject Classification** Theory of computation → Modal and temporal logics

**Keywords and phrases** Temporal logic, first-order logic, knowledge-representation, infinite-state systems

**Digital Object Identifier** 10.4230/LIPIcs.TIME.2025.2

**Category** Invited Talk

## 1 Introduction

*Linear Temporal Logic* (LTL) [16] is one of the most common formalisms to express temporal properties of systems in many fields including *formal verification* and *artificial intelligence*. In its classic form, LTL is a propositional modal logic interpreted over *infinite* linear orders or *words*, although recently interest has risen in the artificial intelligence field for $\mathsf{LTL}_\mathsf{f}$ [3], *i.e.* LTL interpreted over *finite* words.

The success of LTL stems from its intuitive syntax and semantics and the existence of many efficient techniques for *reasoning* about the logic. Indeed, although *satisfiability* of LTL formulas is PSPACE-complete [21], many efficient techniques are known [9, 12, 22], and the same can be said for *model checking* [15].

However, in many scenarios, the *propositional* nature of LTL poses some limits to its applicability. Consider the following classic example:

$$\mathsf{G}(req \rightarrow \mathsf{F}grant)$$

The above formula states that at any given moment (G), if a *request* is received, then *eventually* (F) an answer is *granted*. This kind of specification is quite common, but it is rarely sufficient to express it in the above way. That is because the above formula makes no connection between the answer that is granted and the request that is being answered. For example, the formula is also satisfied by a single answer after many requests.

What one would really like to express is that *each* request gets *its own* answer. One may wonder what the *identity* of requests consist in. Let us suppose requests have unique identifiers that last for all the execution of the system, and suppose that, instead of propositions, we can use the $req(r)$ predicate to tell that the request $r$ has been requested, and $grant(r)$ to tell that $r$ has been answered. Then, we can write the following:

$$\forall r \,.\, \mathsf{G}(req(r) \rightarrow \mathsf{F}grant(r))$$

The above specification is a *sentence* in *first-order linear temporal logic* (FOLTL). FOLTL combines the usual *temporal operators* from LTL with classic *first-order logic*, obtaining a remarkably expressive and sophisticated language. In what follows we introduce the semantics of this language (Section 2), its major computational trade-offs (Section 3), and we describe some current trends (Section 4).

## 2 Semantics

In first-order logic, once the *signature* $\Sigma$ is fixed (*i.e.* the set of constant, predicate and function symbols used), sentences (*i.e.* formulas with no free variables) are interpreted over $\Sigma$-structures that interpret the signature's symbols. In FOLTL, sentences are interpreted over *sequences* of $\Sigma$-structures, representing the evolution over time of the symbols' interpretation. Classically, FOLTL has been studied over many different kind of sequences or *linear orders* of $\Sigma$-structures: finite or infinite, discrete or *dense*, *etc.*, with most classic results holding independently of the nature of the underlying linear order [5]. Here, for simplicity, let us assume *discrete* linear orders, either finite or infinite.

When formally defining the semantics of the *satisfaction* of FOLTL formulas, one immediately encounters a non-trivial question about the meaning of *existence* of the objects of first-order quantification. To see what this means, suppose we are modeling the human resources of a company, and we care about the feelings of our employees. We may write a sentence like the following.

$$\psi := \forall x.\mathsf{G}happy(x) \rightarrow \mathsf{G}\forall x.happy(x)$$

The above sentence is an instance of a scheme called the *Barcan formula* [14]. In this case, it is saying that, in the company, if everyone is happy every day, then every day everyone is happy. One may consider $\psi$ to be obviously a *valid* FOLTL sentence or not, depending on two different views:

1. the sentence is valid because the two mentions of *"everyone"* in the above phrase refer to the same set of people, *i.e.* the domain of the two universal quantifiers is the same; or
2. the sentence is not valid because who is *"everyone"* today may include or not include people that will be absent or present tomorrow, *i.e.* the domain of the two universal quantifiers may differ.

Depending on our choice of point of view, we can identify a different *semantics* for FOLTL:

1. *eternalist* or *constant-domain* semantics: every and all objects in the quantification domain always exist at any point in time, as the domain *does not change*;
2. *presentist* or *varying-domain* semantics: objects in the domain pop up into existence at some time and cease to exist at some later time, as the domain *varies over time*.

In the eternalist semantics, the Barcan sentence is valid and we can swap the universal quantifier with the *always* temporal operator (or, equivalently, the existential quantifier with the *eventually* operator). In the presentist semantics, we cannot do that. The choice of semantics affects the semantics of purely first-order sentences as well. Consider for example the following sentence:

$$\forall x.happy(x) \rightarrow happy(\mathsf{Nik})$$

This is an instance of the *universal instantiation* axiom which is *valid* in classical first-order logic for any formula in the place of $happy(x)$. However, in the varying-domain semantics the sentence is not valid anymore, because the constant Nik may refer to somebody that is not *currently* in the domain (*e.g.* because he still has to be hired) and therefore *may* be not happy. An axiomatization of FOLTL in both semantics can be found in McArthur's book [14].

Note that, in modeling terms, the *constant-domains* semantics is the most general: one can simulate the varying-domain semantics in a constant-domain sentence by introducing an *existence predicate* $exists(x)$ guarding all occurrences of quantifiers.

## 3 Computational trade-offs

Depending on the signature $\Sigma$ and the $\Sigma$-*theory* we consider, FOLTL can be extremely expressive. Of course, this expressiveness is payed in terms of *tractability*, as both *satisfiability* and *validity* are *undecidable* and *not even semi-decidable* [5]. This can be seen easily, for example, by reducing the *recurrent tiling* problem [23] to FOLTL satisfiability.

Unfortunately, research has shown that *decidable* fragments of FOLTL are rare and far apart [5]. A classic example of decidable fragment of FOLTL is the *monodic* fragment. A sentence is *monodic* if any *temporal* subformula has at most *one free variable*. For example:

$$\forall x[p(x) \rightarrow Gp(x)]$$

is a monodic sentence, as is the *Barcan* sentence discussed above. Instead:

$$\forall xy[r(x, y) \rightarrow Gr(x, y)]$$

is *not* monodic. The *monodic* fragment is made of *relational* monodic sentences *with no* equality symbol, and can be proved to be decidable over any kind of linear order through a reduction to Büchi's decidability result for monadic second-order logic [5] or, for discrete linear orders, via *first-order automata* [7].

It is clear that the major restriction of the monodic fragment is the inability of transferring *relational* information across time. As this is a major limitation for the modeling of many scenarios, one may wonder whether more expressiveness can be recovered by accepting *semi-decidability*. After all, many decades of research in the automated reasoning community have proven that semi-decidable problems *can* be addressed in practice is suitably effective semi-decision procedures are found. An example is that of *constrained Horn clauses*, a semi-decidable fragment of first-order logic effectively solved in practice by *property-directed reachability* techniques [11].

In this vein, one may prove (*e.g.*, again via first-order automata [7]) that if one starts from a combination of *decidable* first-order logic fragment and theory (*e.g.* the *two-variable fragment* [10] or many decidable theories employed in *satisfiability modulo theories* (SMT) [2]), then FOLTL over *finite words* is *semi-decidable*.

## 4 Recent trends

Recent trends go in the mentioned direction of accepting computational trade-offs in exchange of more expressive power, unlocking the usage of (well-behaved fragments of) FOLTL in the specification of infinite-state systems.

An example of work in this direction is $\mathsf{LTL_f}$ *modulo theories* ($\mathsf{LTL_f^{MT}}$) [6, 8], a recently-introduced fragment of FOLTL interpreted over *finite* words that, although semi-decidable, has a decision procedure effectively implementable in terms of modern SMT solvers.

$\mathsf{LTL_f^{MT}}$ poses semantic and syntactic restrictions to FOLTL. Semantically, it is interpreted, in the *constant-domain* semantics, over finite linear orders where the interpretation of predicates and function symbols is *rigid*, *i.e.* arbitrary but fixed in time, and only the

interpretation of *constants* is allowed to change. Syntactically, it forbids the alternation of quantifiers and temporal operators but allows the usage of a *lookahead* operator that, applied to a constant, designates the value of the constant at the next time step. For example:

$$(\rhd\, a = a + 1)\ \mathsf{U}\ (a = 42)$$

is a $\mathsf{LTL}_\mathsf{f}^\mathsf{MT}$ sentence saying that the constant $a$ increments by one at each time step until it reaches the value 42. One may rewrite such a sentence in pure FOLTL as follows[1]:

$$(\exists x.[\mathsf{X}(x = a) \wedge x = a + 1])\ \mathsf{U}\ (a = 42)$$

The $\mathsf{LTL}_\mathsf{f}^\mathsf{MT}$ logic has been defined over finite words because its semi-decidability (under the conditions mentioned in the previous section) cannot be proven in general if infinite words are involved. However, some *decidability* conditions have been identified [8] which hold for infinite words as well (*i.e.* $\mathsf{LTL}^\mathsf{MT}$). These results complement the classic ones on the monodic fragment: $\mathsf{LTL}_\mathsf{f}^\mathsf{MT}$ *can* be translated into monodic FOLTL sentences, but the classic decidability results mentioned above hold for *rigid constants* and *non-rigid predicates*, which is exactly the opposite semantic setting than $\mathsf{LTL}_\mathsf{f}^\mathsf{MT}$, in addition to the fact that $\mathsf{LTL}_\mathsf{f}^\mathsf{MT}$ allows the equality symbol.

The structure of the logic has been designed in order to have its satisfiability problem being easily reduced into a sequence of SMT calls that can be solved by standard solvers, obtained by an SMT encoding of a suitable extension of Reynolds' tree-shaped tableau for LTL [9, 18]. Performance are promising in practice, although more work is still needed to exploit the full potential of the approach.

Recently, interest has sparked about a task that goes beyond satisfiability and validity, *i.e.* *reactive synthesis*. This is the task of synthesizing a *controller* that can ensure the satisfaction of a temporal formula independently from the actions of an external antagonistic environment. Reactive synthesis for propositional LTL and $\mathsf{LTL}_\mathsf{f}$ is already a hard problem (2EXPTIME-complete [4, 17]), and is of course undecidable for FOLTL and for $\mathsf{LTL}_\mathsf{f}^\mathsf{MT}$ as well.

However, progress has been made on addressing the problem in practice. In particular, *equirealizable* Boolean abstractions have been found for $\mathsf{LTL}^\mathsf{MT}$ *without lookaheads* [20] which can be given to propositional LTL synthesizers, whose produced strategies can be mapped back to a strategy for the original $\mathsf{LTL}^\mathsf{MT}$ sentence. Then, the approach has been extended to $\mathsf{LTL}^\mathsf{MT}$ with lookaheads by a counterexample-guided abstraction refinement procedure [19], although unlocking full potential of this technique requires manual intervention in the loop.

## 5 Conclusions

The field of first-order temporal logics intersects with first-order modal logics, knowledge representation, temporal *description logics* [1, 13], satisfiability modulo theories, and many other corners of computer science, in a fascinating and intricate web of connections.

Because of its discouraging computational behavior, FOLTL has been studied during the decades in mostly theoretical terms and with a focus on modeling and knowledge representation rather than reasoning. Nevertheless, recent work has highlighted the possibility of dealing in practice with reasoning tasks over expressive fragments of FOLTL, including *reactive synthesis*. Progress in this field is encouraging and this short abstract also wants to be a call to action for the community to invest resources in this promising direction.

[1] The semantics given in [6] needs to be slightly tweaked for this translation to work in general.

## References


1 Alessandro Artale and Enrico Franconi. Temporal description logics. In Michael Fisher, Dov M. Gabbay, and Lluís Vila, editors, *Handbook of Temporal Reasoning in Artificial Intelligence*, volume 1 of *Foundations of Artificial Intelligence*, pages 375–388. Elsevier, 2005. doi:10.1016/S1574-6526(05)80014-8.

2 Clark W. Barrett, Roberto Sebastiani, Sanjit A. Seshia, and Cesare Tinelli. Satisfiability modulo theories. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, pages 825–885. IOS Press, 2009. doi:10.3233/978-1-58603-929-5-825.

3 Giuseppe De Giacomo and Moshe Y. Vardi. Linear temporal logic and linear dynamic logic on finite traces. In *Proceedings of 23rd IJCAI*, pages 854–860, 2013. URL: http://www.aaai.org/ocs/index.php/IJCAI/IJCAI13/paper/view/6997.

4 Giuseppe De Giacomo and Moshe Y. Vardi. Synthesis for LTL and LDL on finite traces. In Qiang Yang and Michael J. Wooldridge, editors, *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 1558–1564. AAAI Press, 2015. URL: http://ijcai.org/Abstract/15/223.

5 D.M. Gabbay, A. Kurucz, F. Wolter, and M. Zakharyaschev. Fragments of first-order temporal logics. In *Many-Dimensional Modal Logics*, volume 148 of *Studies in Logic and the Foundations of Mathematics*, pages 465–545. Elsevier, 2003. doi:10.1016/S0049-237X(03)80012-5.

6 Luca Geatti, Alessandro Gianola, and Nicola Gigante. Linear temporal logic modulo theories over finite traces. In *Proceedings of the 31st IJCAI*, pages 2641–2647, 2022. doi:10.24963/ijcai.2022/366.

7 Luca Geatti, Alessandro Gianola, and Nicola Gigante. First-order automata. In Toby Walsh, Julie Shah, and Zico Kolter, editors, *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*, pages 14940–14948. AAAI Press, 2025. doi:10.1609/AAAI.V39I14.33638.

8 Luca Geatti, Alessandro Gianola, Nicola Gigante, and Sarah Winkler. Decidable fragments of $\mathrm{ltl}_{\mathrm{f}}$ modulo theories. In *Proceedings of the 26th ECAI*, volume 372 of *Frontiers in Artificial Intelligence and Applications*, pages 811–818. IOS Press, 2023. doi:10.3233/FAIA230348.

9 Luca Geatti, Nicola Gigante, Angelo Montanari, and Gabriele Venturato. SAT meets tableaux for linear temporal logic satisfiability. *J. Autom. Reason.*, 68(2):6, 2024. doi:10.1007/S10817-023-09691-1.

10 Erich Grädel, Phokion G. Kolaitis, and Moshe Y. Vardi. On the decision problem for two-variable first-order logic. *Bull. Symb. Log.*, 3(1):53–69, 1997. doi:10.2307/421196.

11 Arie Gurfinkel and Nikolaj S. Bjørner. The science, art, and magic of constrained horn clauses. In *Proceedings of the 21st International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, pages 6–10. IEEE, 2019. doi:10.1109/SYNASC49474.2019.00010.

12 Jianwen Li, Yinbo Yao, Geguang Pu, Lijun Zhang, and Jifeng He. Aalta: an LTL satisfiability checker over infinite/finite traces. In Shing-Chi Cheung, Alessandro Orso, and Margaret-Anne D. Storey, editors, *Proc. of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 731–734. ACM, 2014. doi:10.1145/2635868.2661669.

13 Carsten Lutz, Frank Wolter, and Michael Zakharyaschev. Temporal description logics: A survey. In Stéphane Demri and Christian S. Jensen, editors, *15th International Symposium on Temporal Representation and Reasoning, TIME 2008, Université du Québec à Montréal, Canada, 16-18 June 2008*, pages 3–14. IEEE Computer Society, 2008. doi:10.1109/TIME.2008.14.

14 Robert P. McArthur. *Tense Logic*. Springer, 1976. doi:10.1007/978-94-017-3219-2.

15 Kenneth L. McMillan. Interpolation and model checking. In Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem, editors, *Handbook of Model Checking*, pages 421–446. Springer, 2018. doi:10.1007/978-3-319-10575-8_14.

16 Amir Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science*, pages 46–57. IEEE Computer Society, 1977. doi:10.1109/SFCS.1977.32.

**17** Amir Pnueli and Roni Rosner. On the synthesis of an asynchronous reactive module. In *16th International Colloquium on Automata, Languages and Programming*, volume 372 of *Lecture Notes in Computer Science*, pages 652–671. Springer, 1989. doi:10.1007/BFB0035790.

**18** Mark Reynolds. A New Rule for LTL Tableaux. In *Proc. of the 7$^{th}$ International Symposium on Games, Automata, Logics and Formal Verification*, volume 226 of *EPTCS*, pages 287–301, 2016. doi:10.4204/EPTCS.226.20.

**19** Andoni Rodriguez, Felipe Gorostiaga, and Cesar Sanchez. Counter example guided reactive synthesis for ltl modulo theories. In *Proceedings of the 37th International Conference on Computer Aided Verification (CAV 2025)*, page to appear, 2025.

**20** Andoni Rodríguez and César Sánchez. Boolean abstractions for realizability modulo theories. In *Proceedings of the 35th CAV*, volume 13966 of *Lecture Notes in Computer Science*, pages 305–328. Springer, 2023. doi:10.1007/978-3-031-37709-9_15.

**21** A. Prasad Sistla and Edmund M. Clarke. The complexity of propositional linear temporal logics. *J. ACM*, 32(3):733–749, 1985. doi:10.1145/3828.3837.

**22** M. Suda and C. Weidenbach. A PLTL-Prover Based on Labelled Superposition with Partial Model Guidance. In *Proc. of the 6th International Joint Conference on Automated Reasoning*, volume 7364 of *LNCS*, pages 537–543. Springer, 2012. doi:10.1007/978-3-642-31365-3_42.

**23** Peter van Emde Boas et al. The convenience of tilings. *Lecture Notes in Pure and Applied Mathematics*, pages 331–363, 1997.