TOWARDS INFINITE-STATE VERIFICATION AND PLANNING WITH LINEAR TEMPORAL LOGIC MODULO THEORIES

Luca Geatti
University of Udine, Italy

Alessandro Gianola
University of Lisbon, Portugal

Nicola Gigante
Free University of Bozen-Bolzano, Italy

TIME 2023
Athens, Greece
September 25, 2023

Linear Temporal Logic

Linear Temporal Logic (LTL) is the most common formalism to specify temporal properties in **formal verification** and **artificial intelligence**.

- **propositional** modal logic interpreted over **infinite** or **finite** traces (LTLf)
- studied since the '70s [Pnu77]
- many efficient reasoning techniques despite the high complexity
- many mature software tools employing it

Limits of propositional logics

The **propositional** nature of LTL and similar logics limits them to **finite-state** systems.

However, many scenarios are difficult or impossible to abstract finitely:

- systems involving **arithmetics**
- systems involving complex and unbounded **data structures**
- systems involving **relational databases**

LTL modulo theories

For this reason, we introduced **LTL modulo theories** (LTL^{MT}) [GGG22]:

- first-order extension of LTL
- propositions are replaced by **first-order sentences** over arbitrary theories, *à la* SMT
- (semi-)decision procedures based on off-the-shelf **SMT solvers**

LTL modulo theories

$\mathrm{LTL}^{\mathrm{MT}}$ is not the first first-order extension of LTL, however:

- many first-order temporal logics have been extensively studied from **theoretical** perspectives but without any practical development (see, *e.g.* [Kon+04])
- others led to practically applicable approaches but support quite **ad-hoc** syntax and semantics (see, *e.g.* [Cim+20])

Our approach is at the same time **theoretically** well-grounded, **general**, and **practically** oriented.

The BLACK reasoner

LTL^{MT} is supported by our **BLACK**[1] temporal reasoning framework:[2]

- a software library and tool for **temporal reasoning** in linear-time logics
- supports LTL and LTL^{MT} in many flavors
- playground for many of our research directions

[1] Bounded LTL sAtisfiability ChecKer
[2] https://www.black-sat.org

Plan of the talk

1. LTL modulo theories
2. Verification of LTLf^{MT} properties
3. Future directions

LTL MODULO THEORIES

Linear Temporal Logic

$$\alpha \, \mathsf{U} \, \beta \qquad \mathsf{X}\alpha \qquad \mathsf{F}\beta \qquad \mathsf{G}\alpha$$

$$\alpha \,\mathsf{U}\, \beta \qquad \mathsf{X}\alpha \qquad \mathsf{F}\beta \qquad \mathsf{G}\alpha$$

β holds somewhere in the future, and α holds everywhere **until** then.

Linear Temporal Logic

$\alpha \,\mathsf{U}\, \beta \qquad \mathsf{X}\alpha \qquad \mathsf{F}\beta \qquad \mathsf{G}\alpha$

α holds at the **next** state.

$$\alpha \, \mathsf{U} \, \beta \qquad \mathsf{X}\alpha \qquad \mathsf{F}\beta \qquad \mathsf{G}\alpha$$

β holds somewhere in the **future** ($\top \, \mathsf{U} \, \beta$)

Linear Temporal Logic

$\alpha \,\mathsf{U}\, \beta \qquad \mathsf{X}\alpha \qquad \mathsf{F}\beta \qquad \mathsf{G}\alpha$

α **always** holds from now ($\neg\mathsf{F}\neg\alpha$).

Infinite and finite traces

LTL can be interpreted over **finite** or **infinite** traces.

- the **infinite-trace** semantics is the historically more studied [Pnu77]
- the **finite-trace** semantics gained attention recently (LTLf) [DV13]
- finite traces are algorithmically much **easier** to deal with
  - *e.g.*, NFAs instead of Büchi automata

Finite traces

The finite-traces semantics is quite different. For example:

- $\mathsf{GX}\top$ is not valid anymore, it is actually **unsatisfiable**
- $\neg\mathsf{X}\phi \not\equiv \mathsf{X}\neg\phi$
- $\mathsf{GF}\phi$ only means ϕ holds at the **last** state

The **weak tomorrow** operator is usually introduced:

$$\widetilde{\mathsf{X}}\phi \equiv \neg\mathsf{X}\neg\phi$$

ϕ holds at the next state, **if it exists**.

Satisfiability

Satisfiability

Is there a state sequence that **satisfies** a given formula ϕ?

Satisfiability

LTL satisfiability is a versatile problem.

- **entailment** and **validity** can be reduced to satisfiability:

$$\begin{aligned} \phi \text{ is valid} \quad & \text{iff} \quad \neg\phi \text{ is unsat.} \\ \phi \supset \psi \quad & \text{iff} \quad \phi \rightarrow \psi \text{ is valid} \end{aligned}$$

- **model-checking** can be reduced to satisfiability:

$$M \models \psi \quad \text{iff} \quad \phi_M \supset \psi$$

- **sanity checking** of specifications is a satisfiability question:
  - unsatisfiable specifications are buggy
  - valid specifications are useless
- **STRIPS planning** can be reduced to LTL satisfiability [May+07]

Data-aware systems

Data-aware systems

Systems that involve the processing and manipulation of **data** taken from an **infinite** domain.

Examples:

- (relational) database-driven systems
- systems involving complex data-structures
- systems involving arithmetics
- any combination of the above!

Data-aware systems are **infinite-state**, leading very easily to **undecidability** of verification, model-checking, satisfiability etc ...

But they are still worth studying!

LTL modulo theories

LTL$^{\mathsf{MT}}$ is our take at the verification of **infinite-state** data-aware systems.

LTL$^{\mathsf{MT}}$ extends LTL by replacing **propositions** with **first-order sentences**.

- symbols can be uninterpreted, or interpreted by arbitrary first-order theories
  - *e.g.*, $+$, $<$ interpreted as integer sum/comparison
- constants, relational/function symbols, *etc.* can be both **rigid** or **non-rigid**
- interpreted over **finite-traces** semantics (see later why)
  - so we actually talk about **LTLf$^{\mathsf{MT}}$**

Examples

$$\mathsf{G}(x = 2y) \qquad (x < y)\,\mathsf{U}\,(y = 0) \qquad \mathsf{G}(x > 5) \land \mathsf{F}(x = 0)$$

$$\mathsf{G}(\exists y(x = 2y))$$

Examples

$$\mathsf{G}(x = 2y) \qquad (x < y)\, \mathsf{U}\, (y = 0) \qquad \mathsf{G}(x > 5) \wedge \mathsf{F}(x = 0)$$

$$\mathsf{G}(\exists y (x = 2y))$$

Examples

$$\mathrm{G}(x = 2y) \qquad (x < y)\,\mathrm{U}\,(y = 0) \qquad \mathrm{G}(x > 5) \wedge \mathrm{F}(x = 0)$$

$$\mathrm{G}(\exists y(x = 2y))$$

Examples

$$\mathsf{G}(x = 2y) \qquad (x < y)\ \mathsf{U}\ (y = 0) \qquad \mathsf{G}(x > 5) \wedge \mathsf{F}(x = 0)$$

$$\mathsf{G}(\exists y (x = 2y))$$

Examples

$$\mathrm{G}(x = 2y) \qquad (x < y)\,\mathrm{U}\,(y = 0) \qquad \mathrm{G}(x > 5) \wedge \mathrm{F}(x = 0)$$

$$\mathrm{G}(\exists y (x = 2y))$$

Examples

$$x = 0 \wedge ((\bigcirc x = x + 1) \mathsf{U}\, x = 42)$$

$$y = 1 \wedge \mathsf{G}(\circledtilde\, y = y + 1 \wedge x = 2y)$$

$$p(0) \wedge \mathsf{G}\forall x(p(x) \to \widetilde{\mathsf{X}} p(x+1)) \wedge \mathsf{F} p(42)$$

Examples

$$x = 0 \wedge ((\bigcirc x = x + 1) \mathsf{U}\, x = 42)$$

$$y = 1 \wedge \mathsf{G}(\circledtilde\, y = y + 1 \wedge x = 2y)$$

$$p(0) \wedge \mathsf{G}\forall x(p(x) \to \widetilde{\mathsf{X}} p(x+1)) \wedge \mathsf{F} p(42)$$

Examples

$$x = 0 \land ((\bigcirc x = x + 1) \,\mathsf{U}\, x = 42)$$

$$y = 1 \land \mathsf{G}(\circledsim y = y + 1 \land x = 2y)$$

$$p(0) \land \mathsf{G}\forall x(p(x) \to \widetilde{\mathsf{X}}p(x + 1)) \land \mathsf{F}p(42)$$

Examples

$$x = 0 \wedge ((\bigcirc x = x + 1) \,\mathsf{U}\, x = 42)$$

$$y = 1 \wedge \mathsf{G}(\widetilde{\bigcirc}\, y = y + 1 \wedge x = 2y)$$

$$p(0) \wedge \mathsf{G}\forall x(p(x) \to \widetilde{\mathsf{X}}p(x+1)) \wedge \mathsf{F}p(42)$$

Where's the catch?

LTL^{MT} is clearly **undecidable**, but:

- with **finite traces** semantics, and over **decidable** first-order theories, it is **semi-decidable**
  - so we actually talk more about **LTLf^{MT}**
- our **semi-decision** procedure always answers **yes** for satisfiable formulas, **may** not terminate for unsatisfiable ones (but sometimes does)
- decidable theories and first-order fragments abound, *e.g.*:
  - linear integer/real **arithmetic** (LIA/LRA)
  - quantifier-free equality and **uninterpreted** functions (QF_EUF)
  - arrays, fixed-size bitvectors, algebraic data types, **floating-point** numbers, *etc.*
  - **effectively propositional** (EPR) logic: $\exists^*\forall^*\phi$
  - **two-variables** first-order logic (FO^2)

Where's the catch?

LTL^{MT} is clearly **undecidable**, but:

- **decidable fragments** of LTLf^{MT} exist (see ECAI '23)
- foundamental concept: **history constraints**
- first-order formulas summarizing the effect of the history on the variables
- a theory has **finite memory** if the possible history constraints are finite (up to T-equivalence)
- decidability follows

Examples:

- purely relational theories
- locally finite theories (*e.g.*, modular arithmetic)
- bounded lookback formulas
- **cosafety** formulas (FX)

How to solve LTLf modulo theories

How do we test satisfiability of $\mathrm{LTLf}^{\mathrm{MT}}$ formulas?

- an **iterative** procedure tests the existence of models of length up to $k \geqslant 0$, for increasing values of k
- given an $\mathrm{LTLf}^{\mathrm{MT}}$ formula ϕ and a k, we build a **purely first-order formula** $\langle\phi\rangle_k$ that is **satisfiable** if and only if there is a model for ϕ of length at most k
- $\langle\phi\rangle_k$ is given to an off-the-shelf SMT solver

VERIFICATION OF LTLf^{MT} PROPERTIES

Knowledge-base driven Dynamic Systems

Which systems can we verify $\mathrm{LTLf}^{\mathrm{MT}}$ formulas on?

Knowledge-base driven Dynamic Systems (KDS):

- infinite-state transition systems

$$D = \langle \mathrm{K}, I(X), C(X), T(X, X'), F(X) \rangle$$

- states are structures over the **first-order theory** K (*e.g.*, integers, reals, EUF, *etc.*)
- $I(X)$, $T(X, X')$, $F(X)$ are arbitrary **first-order** formulas over the theory K
  - **initial** states satisfying $I(X)$
  - **final** states satisfying $F(X)$
  - **transition relation** expressed by $T(X, X')$

Verification of KDSs

Let D be a KDS and ϕ an LTLf^{MT} formula:

- all the executions of a KDS D can be **represented** by an LTLf^{MT} formula ψ_D
- **model-checking** of ϕ over D reduces to **satisfiability** of:

$$\gamma \equiv \psi_D \wedge \neg\phi$$

- if γ is **satisfiable**, the specification does not hold and the model is a counterexample
- if γ is **unsatisfiable**, the specification is valid over D

Some experiments

That's cool, but **does it work**?

- everything here is **undecidable**

Some experiments

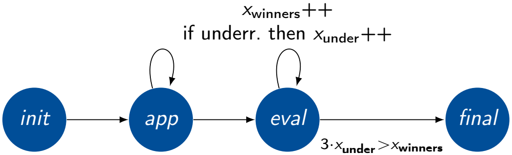
That's cool, but **does it work**?

- everything here is **undecidable**
- but. . .

Some early experiments

Test setting:

- simulation of a company hiring process
- nondeterministic transitions:
  - dependent on arithmetic constraints
  - acting on unbounded relational data
- minimal length of the counterexamples dependent over scalable parameter N
- two modelings of the same system:
  - P_1 employs arithmetic constraints
  - P_2 avoids arithmetics, simulates constraints by other means
- two different properties for each variant



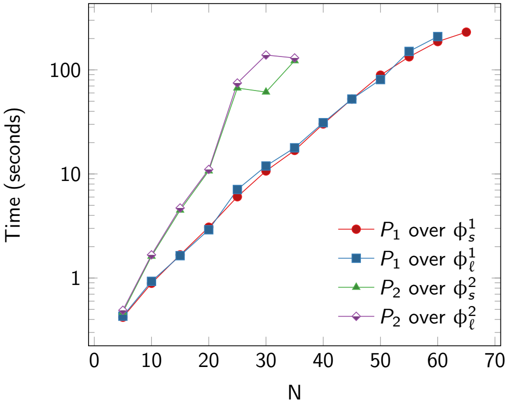
$$\phi_s^1 \equiv \mathsf{G}(x_{state} = \mathit{final} \rightarrow 2x_{\text{under}} > x_{\text{winners}})$$

$$\phi_\ell^1 \equiv \mathsf{G}\begin{pmatrix} x_{state} = \mathit{app} \rightarrow \\ F(x_{\text{state}} = \mathit{final} \wedge 2x_{\text{under}} > x_{\text{winners}}) \end{pmatrix}$$

Some early experiments

Results:

- 5 minutes timeout reached at $N = 70$
- **exponential** growth
  - but could be much worse,
    the problem is **undecidable**!
- liveness property not harder than the safety one
- system with **explicit arithmetics** faster to verify



FUTURE DIRECTIONS

Data-aware planning

Automated planning languages and techniques are currently limited in **expressiveness**:

- propositional variables
- sometimes numbers
- sometimes temporal constraints

What if my agent needs to reason on more **complex domains**?

- interaction with relational databases (*e.g.*, in a warehouse)
- manipulation of data structures (lists, trees, graphs, ...)
- manipulation of **ontologies**

Data-aware planning

One can reduce planning in **complex domains** as LTLf$^{\mathrm{MT}}$ **satisfiability**.

- formula model $\rightarrow$ plan

One may reduce **FOND** planning in such domains as LTLf$^{\mathrm{MT}}$ **synthesis**:

- strategy $\rightarrow$ policy

Long term goal: a unified perspective on infinite-state verification and data-aware planning.

Future directions

Other future directions:

- find an SMT **encoding** of the history constraints
- find more **efficient** LTLf$^{\text{MT}}$ fragments (not necessarily decidable)
- further implementation developments:
  - better integration with the underlying SMT solvers
  - general support for any backend-supported theory (bitvectors, arrays, **floating-point**, ADTs, *etc.*)
  - systems modeling language
- embedding of **temporal description logics**
- reactive **synthesis**
- is there a corresponding **automaton** model?

THANK YOU

REFERENCES

References

[Cim+20] Alessandro Cimatti, Alberto Griggio, Enrico Magnago, Marco Roveri, and Stefano Tonetta. "SMT-based satisfiability of first-order LTL with event freezing functions and metric operators." In: *Inf. Comput.* 272 (2020), p. 104502. DOI: 10.1016/j.ic.2019.104502.

[DV13] Giuseppe De Giacomo and Moshe Y. Vardi. "Linear Temporal Logic and Linear Dynamic Logic on Finite Traces." In: *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*. Ed. by Francesca Rossi. IJCAI/AAAI, 2013, pp. 854–860.

[GGG22] Luca Geatti, Alessandro Gianola, and Nicola Gigante. "Linear Temporal Logic Modulo Theories over Finite Traces." In: *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*. ijcai.org, 2022, pp. 2641–2647. DOI: 10.24963/ijcai.2022/366.

[Kon+04] Roman Kontchakov, Carsten Lutz, Frank Wolter, and Michael Zakharyaschev. "Temporalising Tableaux." In: *Stud Logica* 76.1 (2004), pp. 91–134. DOI: 10.1023/B:STUD.0000027468.28935.6d.

References

[May+07] Marta Cialdea Mayer, Carla Limongelli, Andrea Orlandini, and Valentina Poggioni. "Linear temporal logic as an executable semantics for planning languages." In: *Journal of Logic, Language and Information* 16.1 (2007), pp. 63–89. DOI: 10.1007/s10849-006-9022-1.

[Pnu77] A. Pnueli. "The Temporal Logic of Programs." In: *Proc. of the 18th Annual Symposium on Foundations of Computer Science*. IEEE Computer Society, 1977, pp. 46–57. DOI: 10.1109/SFCS.1977.32.