First-Order Linear Temporal Logic for Testing Distributed Protocols

José João Ferreira, Nuno Policarpo, José Fragoso Santos, Alcino Cunha and Alessandro Gianola

INESC-ID / Instituto Superior Técnico,
University of Lisbon, Portugal

OVERLAY 2025 - Bologna, Italy, October 26
Automata, formal languages and logic

Introduction

The importance of distributed systems and protocols

Introduction

The importance of distributed systems and protocols

- large-scale social networks

Introduction

The importance of distributed systems and protocols

- large-scale social networks
- financial infrastructures

Introduction

The importance of distributed systems and protocols

- large-scale social networks
- financial infrastructures
- cloud computing and storage platforms

Introduction

The importance of distributed systems and protocols

- large-scale social networks
- financial infrastructures
- cloud computing and storage platforms
- blockchain networks

Introduction

The importance of distributed systems and protocols

- large-scale social networks
- financial infrastructures
- cloud computing and storage platforms
- blockchain networks

. . .

The need for correctness: from algorithms to implementations

Examples of bugs found in implementations

- FaB Paxos [1]
- Raft [2]
- Tendermint [3]
- Gasper [4]

[1] I. Abraham, G. Gueta, D. Malkhi, L. Alvisi, R. Kotla, and J. Martin, "Revisiting fast practical byzantine fault tolerance," *CoRR*, vol. abs/1712.01367, 2017.

[2] C. Jensen, H. Howard, and R. Mortier, "Examining Raft's behaviour during partial network failures," in *Proceedings of the 1st Workshop on High Availability and Observability of Cloud Systems*, ACM, 2021, pp. 11–17.

[3] C. Cachin and M. Vukolic, "Blockchain consensus protocols in the wild," *CoRR*, vol. abs/1707.01873, 2017.

[4] J. Neu, E. N. Tas, and D. Tse, "Ebb-and-flow protocols: A resolution of the availability-finality dilemma," in *42nd IEEE Symposium on Security and Privacy*, IEEE, 2021, pp. 446–465.

How can we validate distributed protocols?

Approaches for validating distributed protocols

- **deductive verification**
  - TLAPS for Paxos or Raft [1]
  - Coq for Raft [2]
  - EPR for Paxos [3]

[1] K. Chaudhuri, D. Doligez, L. Lamport, and S. Merz, "Verifying safety properties with the TLA+ proof system," in *Proceedings of the 5th International Joint Conference on Automated Reasoning*, Springer, 2010, pp. 142–148.

[2] J. R. Wilcox et al., "Verdi: A framework for implementing and formally verifying distributed systems," in *Proceedings of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation*, ACM, 2015, pp. 357–368.

[3] O. Padon, G. Losa, M. Sagiv, and S. Shoham, "Paxos made EPR: decidable reasoning about distributed protocols," *Proceedings of the ACM on Programming Languages*, vol. 1, no. OOPSLA, 108:1–108:31, 2017.

Approaches for validating distributed protocols

- **model checking**
  - Alloy for Chord [1]
  - TLC for Pastry [2]

[1] P. Zave, "Using lightweight modeling to understand Chord," *Comput. Commun. Rev.*, vol. 42, no. 2, pp. 49–57, 2012.

[2] N. Azmy, S. Merz, and C. Weidenbach, "A machine-checked correctness proof for Pastry," *Science of Computer Programming*, vol. 158, pp. 64–80, 2018.

Approaches for validating distributed protocols

- **fuzzing**
  - Mallory for Raft [1]
  - Chronos for Zab and GHOST [2]

[1] R. Meng, G. Pîrlea, A. Roychoudhury, and I. Sergey, "Greybox fuzzing of distributed systems," in *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, ACM, 2023, pp. 1615–1629.

[2] Y. Chen, F. Ma, Y. Zhou, M. Gu, Q. Liao, and Y. Jiang, "Chronos: Finding timeout bugs in practical distributed systems by deep-priority fuzzing with transient delay," in *IEEE Symposium on Security and Privacy*, 2024, pp. 1939–1955.

Approaches for validating distributed protocols

- **monitoring**
  - TLA+ for 2PC [1]
  - TLA+ for Raft [2]
  - ATL for Chord [3]

[1] H. Cirstea, M. A. Kuppe, B. Loillier, and S. Merz, "Validating traces of distributed programs against TLA+ specifications," in *Proceedings of the 22nd Int. Conf. on Software Eng. and Formal Methods*, Springer, 2024, pp. 126–143.

[2] H. Howard, M. A. Kuppe, E. Ashton, A. Chamayou, and N. Crooks, "Smart casual verification of the confidential consortium framework," in *22nd USENIX Symposium on Networked Systems Design and Impl.*, 2025, pp. 259–276.

[3] N. Policarpo, J. F. Santos, A. Cunha, J. Leitão, and P. Á. Costa, "Specifying distributed hash tables with allen temporal logic," in *13th IEEE/ACM International Conference on Formal Methods in Software Engineering*, IEEE, 2025, pp. 63–73.

Online vs offline monitoring

First-Order Linear Temporal Logic

First-Order Linear Temporal Logic

Why is FOLTL suited for...

First-Order Linear Temporal Logic

Why is FOLTL suited for...

- ...distributed protocols?

First-Order Linear Temporal Logic

Why is FOLTL suited for...

- ...distributed protocols?
- ...online monitoring?

First-Order Linear Temporal Logic

Why is FOLTL suited for...

- ...distributed protocols?
- ...online monitoring?
- ...offline monitoring?

First-Order Linear Temporal Logic

Why is FOLTL suited for distributed protocols?

- properties are concerned with time and data
- LTL is not expressive enough
- with past operators, specifications become more natural

First-Order Linear Temporal Logic

Why is FOLTL suited for online monitoring?

- (semi-)decidable fragments of FOLTL and tools
  - TDL using jUnit^{RV} [1]
  - ALTL$_f$ using ADATool [2]
  - LTL$_f^{MT}$ using BLACK [3]
- SMT solvers, tableaux procedures, and automata-based methods

[1] N. Decker, M. Leucker, and D. Thoma, "Monitoring modulo theories," *Int. J. Softw. Tools Technol. Transf.*, vol. 18, no. 2, pp. 205–225, 2016.

[2] P. Felli, M. Montali, F. Patrizi, and S. Winkler, "Monitoring arithmetic temporal properties on finite traces," in *Proceedings of the 37th AAAI Conference on Artificial Intelligence*, AAAI Press, 2023, pp. 6346–6354.

[3] M. Faella and G. Parlato, "A unified automata-theoretic approach to ltl$_f$ modulo theories," in *Proceedings of the 27th European Conference on Artificial Intelligence*, IOS Press, 2024, pp. 1254–1261.

First-Order Linear Temporal Logic

Why is FOLTL suited for offline monitoring?

- applicable, though largely unexplored
- no need for SAT tools
- temporal reasoning, quantification, and predicates

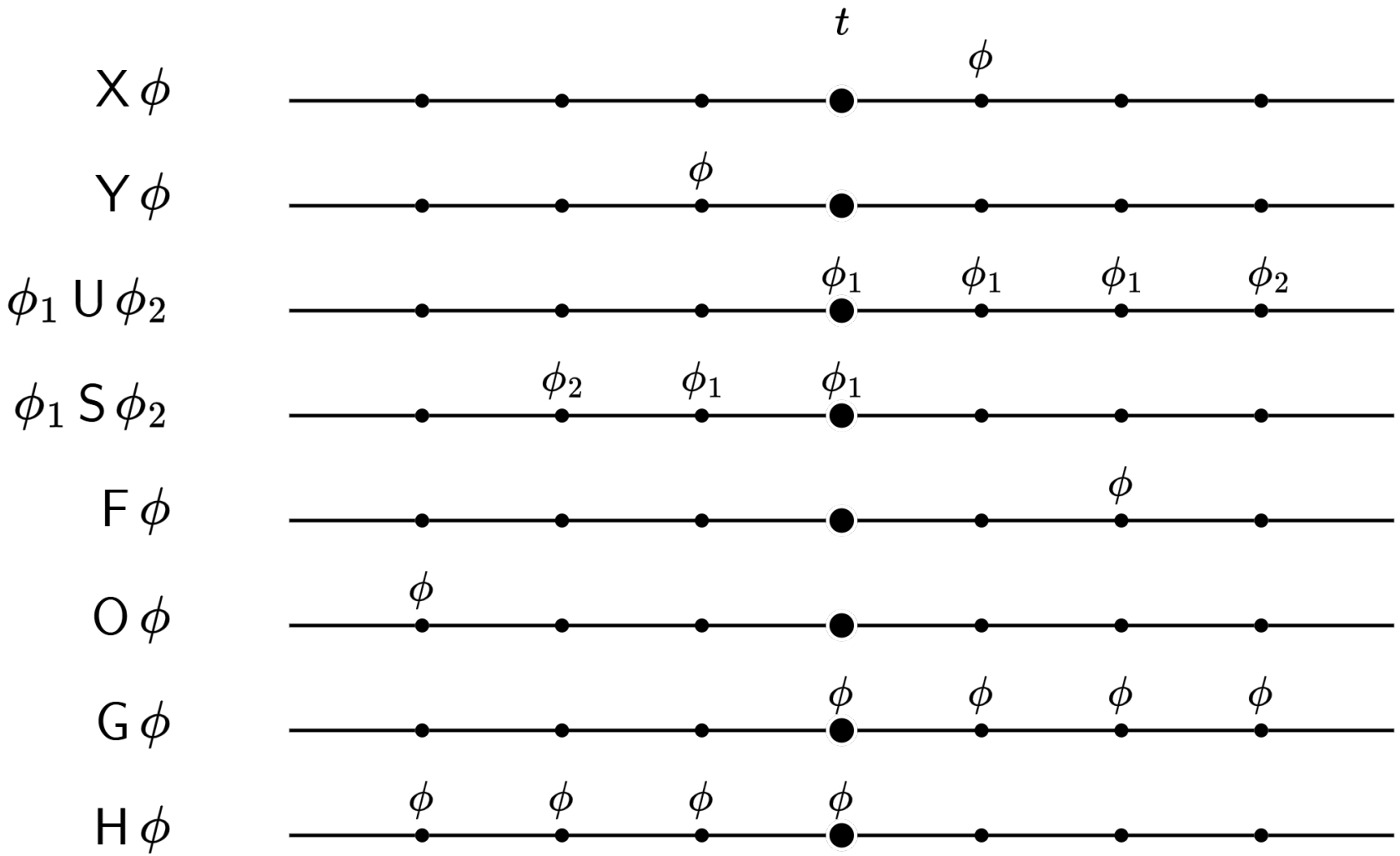
First-Order Linear Temporal Logic

Syntax

$$t := c \mid x \mid f(t_1, \ldots, t_n)$$

$$\begin{aligned}\phi := p(t_1, \ldots, t_n) \mid t_1 = t_2 \mid \neg\phi \mid \phi \vee \phi \mid \exists x \,.\, \phi \\ \mid \mathsf{X}\,\phi \mid \mathsf{Y}\,\phi \mid \phi\,\mathsf{U}\,\phi \mid \phi\,\mathsf{S}\,\phi\end{aligned}$$

Semantics



Two-Phase Commit (2PC)

2PC

- **prepare**(n, t):
  the coordinator requests node n to prepare for transaction t
- **v-commit**(n, t) / **v-abort**(n, t):
  node n votes to commit/abort transaction t
- **i-commit**(t) / **i-abort**(t):
  the coordinator instructs all nodes to commit/abort transaction t

2PC

(LIVENESS) **Eventual decision**: if the coordinator requests any node n to prepare for any transaction t, all nodes will eventually be instructed to commit or abort that transaction

$$\forall\, n, t\,.\, \mathsf{G}\Big(\mathbf{prepare}(n, t) \Rightarrow \mathsf{F}\big(\mathbf{i\text{-}commit}(t) \vee \mathbf{i\text{-}abort}(t)\big)\Big)$$

2PC

(SAFETY) **Atomic commitment**: if the coordinator instructs all nodes to commit any transaction t, then all nodes n must have voted to commit that transaction; if it instructs to abort t, then at least one node n must have voted to abort it

$$\forall\, t\,.\, \mathsf{G}\Big(\big(\textbf{i-commit}(t) \Rightarrow \forall\, n\,.\, \mathsf{O}\, \textbf{v-commit}(n, t)\big) \\ \wedge\, \big(\textbf{i-abort}(t) \Rightarrow \exists\, n\,.\, \mathsf{O}\, \textbf{v-abort}(n, t)\big)\Big)$$

Distributed Hash Table (DHT)

DHT

- **b-store**$_z(n, k, v)$:
  the client requests node n to store the key-value pair $(k,\ v)$
- **e-store**$_z(n')$:
  the operation completes at node n'
- **b-lookup**$_z(n, k)$:
  the client requests the value for key k from node n
- **e-lookup**$_z(n', v)$:
  the operation completes at the node n' storing $(k,\ v)$ and returns v

DHT

(LIVENESS) **Operation termination**: every store or lookup operation eventually completes, that is, for any operation z, once it starts, it will eventually end

$$\forall\, z\,.\, \mathsf{G}\Big(\big(\textbf{b-store}_z(\ldots) \Rightarrow \mathsf{F}\,\textbf{e-store}_z(-)\big) \\ \wedge\ \big(\textbf{b-lookup}_z(\ldots) \Rightarrow \mathsf{F}\,\textbf{e-lookup}_z(\ldots)\big)\Big)$$

DHT

(SAFETY) **Lookup consistency**: if a lookup z obtains a value v for a given key k, then the (k, v) pair was previously written by a store operation

$$\forall\, k, v\,.\, \mathsf{G}\Big(\forall\, z\,.\big(\textbf{b-lookup}_z(-, k) \wedge \mathsf{F}\, \textbf{e-lookup}_z(-, v)\big) \\ \Rightarrow \mathsf{O}\, \textbf{b-store}(-, k, v)\Big)$$

Discussion and Future Work

FOLTL modeling style for DHT

FOLTL modeling style for DHT

- operations modeled with paired begin-end events rather than single predicates spanning multiple time points

Discussion and Future Work

FOLTL modeling style for DHT

- operations modeled with paired begin-end events rather than single predicates spanning multiple time points
- contrasts with 2PC example, showing FOLTL's flexibility

FOLTL modeling style for DHT

- operations modeled with paired begin-end events rather than single predicates spanning multiple time points
- contrasts with 2PC example, showing FOLTL's flexibility
- fragments with syntactic variations may suit protocols with continuous operations and clear input–output distinction

Discussion and Future Work

FOLTL for online monitoring

Discussion and Future Work

FOLTL for online monitoring

- general FOLTL is undecidable for satisfiability checking, but some fragments are (semi-)decidable

Discussion and Future Work

FOLTL for online monitoring

- general FOLTL is undecidable for satisfiability checking, but some fragments are (semi-)decidable
- satisfiability tools (e.g., Alloy, BLACK) could be adapted for runtime trace validation

Discussion and Future Work

FOLTL for online monitoring

- general FOLTL is undecidable for satisfiability checking, but some fragments are (semi-)decidable
- satisfiability tools (e.g., Alloy, BLACK) could be adapted for runtime trace validation
- **challenge:** adapt solvers for online monitoring

Discussion and Future Work

FOLTL for offline monitoring

Discussion and Future Work

FOLTL for offline monitoring

- may not handle checking liveness properties

Discussion and Future Work

FOLTL for offline monitoring

- may not handle checking liveness properties
- domains are trace-bounded, so it avoids satisfiability checking

Discussion and Future Work

FOLTL for offline monitoring

- may not handle checking liveness properties
- domains are trace-bounded, so it avoids satisfiability checking
- monitoring complete traces is decidable and efficient

Discussion and Future Work

FOLTL for offline monitoring

- may not handle checking liveness properties
- domains are trace-bounded, so it avoids satisfiability checking
- monitoring complete traces is decidable and efficient
- **challenge:** build scalable monitors for quantifier-heavy specifications on large traces

Discussion and Future Work

FOLTL for testing distributed protocols

challenges:

Discussion and Future Work

FOLTL for testing distributed protocols

challenges:

- identify useful FOLTL fragments for online/offline monitoring

Discussion and Future Work

FOLTL for testing distributed protocols

challenges:

- identify useful FOLTL fragments for online/offline monitoring
- instrument distributed systems to collect traces

Discussion and Future Work

FOLTL for testing distributed protocols

challenges:

- identify useful FOLTL fragments for online/offline monitoring
- instrument distributed systems to collect traces
- design a monitor to check trace validation against FOLTL formulas

Thank you

First-Order Linear Temporal Logic for Testing Distributed Protocols

José João Ferreira, Nuno Policarpo, José Fragoso Santos, Alcino Cunha and Alessandro Gianola

INESC-ID / Instituto Superior Técnico,
University of Lisbon, Portugal

OVERLAY 2025 - Bologna, Italy, October 26
Automata, formal languages and logic