FIRST-ORDER AUTOMATA

Nicola Gigante
Free University of Bozen-Bolzano, Italy

iFM^2
April 19, 2024
Udine, Italy

Infinite-state specification and verification

Model-checking and other verification techniques for **finite-state** systems have been extremely successful in the past decades.

However, many real-world scenarios require reasoning over **infinite-state** systems.

Infinite-state specification and verification

There are **many** infinite-state formalisms around.

- timed automata [AD94]
- hybrid automata [Alu+92]
- recursive state machines [Alu+05; BMP10]
- visibly pushdown languages [AM04]
- operator-precedence languages [Dro+17]
- FIFO machines [BZ83]
- counter machines [Woj99]
- Petri nets [Mur89; JK09]
- data-aware systems [DLV19; CGM13; Ghi+23]
- automata over infinite alphabets [Seg06; IX19]
- register automata [KF94]

Infinite-state specification and verification

And many **logics** to talk about them.

- precedence oriented temporal logic [CMP22]
- metric temporal logics [Koy90; LVR22; AH94]
- temporal logics with concrete domains [DQ21]
- temporal logics with arithmetics [Fel+23; Cim+20]
- separation logic [CYO01; BK18; Man20]
- constrained horn clauses [GB19]
- logics on data words [Boj+06; DLN07]
- signal temporal logic [MN04]
- first-order temporal logics [Kon+04]

Infinite-state specification and verification

And many **logics** to talk about them.

- precedence oriented temporal logic [CMP22]
- metric temporal logics [Koy90; LVR22; AH94]
- temporal logics with concrete domains [DQ21]
- temporal logics with arithmetics [Fel+23; Cim+20]
- separation logic [CYO01; BK18; Man20]
- constrained horn clauses [GB19]
- logics on data words [Boj+06; DLN07]
- signal temporal logic [MN04]
- **first-order temporal logics** [Kon+04]

First-order Temporal Logic

First-order temporal logics are expressive formalisms to specify properties of infinite-state systems:

- badly undecidable, however;
- used mostly in really tailored fragments or in **knowledge representation** applications where reasoning is not the main task;

Question

Can we make them **practical**, as a specification language for **verification** tasks, while maintaining a substantial fraction of its **expressive power**?

First-order automata

In the finite-state setting, **automata** have played the crucial role of the algorithmic powerhorse of verification and model checking.

Question

Can we define a similar tool for **first-order** temporal logic?

Table of contents

High-level overview of the talk:

- recap on propositional and first-order temporal logics
- past results: LTL modulo theories
- current directions: first-order automata
- conclusions

FIRST-ORDER TEMPORAL LOGICS

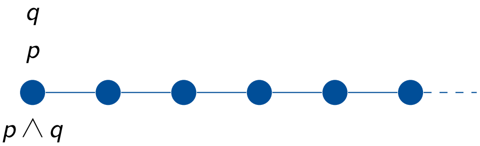
Linear Temporal Logic

Let us recap the main concepts of Linear Temporal Logic:

- modal logic interpreted over **discrete linear orders**
- traditionally, **infinite** linear orders
- recently, **finite** models gained attention [DV13]

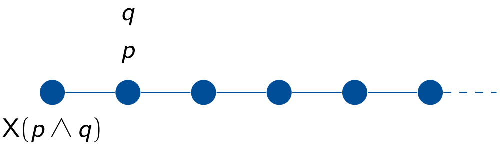
Linear Temporal Logic

Linear Temporal Logic

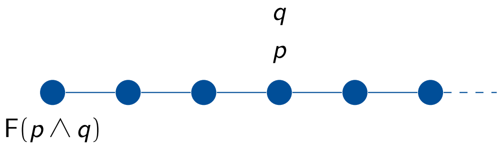


p and q hold

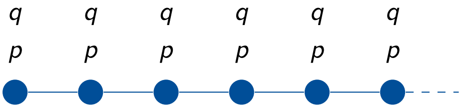
Linear Temporal Logic



tomorrow p and q hold



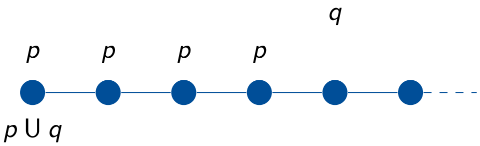
eventually p and q hold



$\mathsf{G}(p \wedge q)$

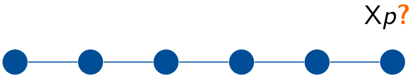
p and q hold **always**

Linear Temporal Logic

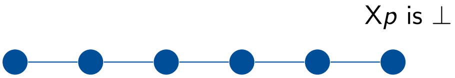


p holds **until** q holds

Linear Temporal Logic over Finite Traces


Xp?

Linear Temporal Logic over Finite Traces



Linear Temporal Logic over Finite Traces

$\widetilde{\mathsf{X}}p \equiv \neg\mathsf{X}\neg p$ is $\top$

weak tomorrow

Linear Temporal Logic over Finite Traces

$\widetilde{\mathsf{X}}\bot$ is $\top$

$\mathsf{X}\top$ is $\bot$

First-order temporal logic

In FOLTL we mix classical first-order logic with LTL temporal operators.

$$\forall x \exists y \, \mathsf{G}\big(p(x) \rightarrow \mathsf{F}(q(x, y))\big) \qquad \mathsf{G}\big(\forall xy \, (p(x, y) \rightarrow \widetilde{\mathsf{X}} p(x, y + 1))\big)$$

$$\forall x \, \mathsf{X} \neg p(x) \land \mathsf{X} \exists x \, p(x)$$

First-order temporal logic

FOLTL is interpreted over **sequences of first-order structures**.

- possibly over **multisorted** signatures;
- first-order features interpreted over the current structure;
- temporal operators to move from one structure to another;
- the **domain** of each sort is arbitrary but **constant** throughout the sequence.

First-order temporal logic

FOLTL is famous for being **highly undecidable**, but, for a fragment F such that:

- the underlying FO fragment is decidable; and
- temporal formulas are **monodic**;
  - only one **free variable** inside temporal operators;
- constants in the signature are considered **rigid**;
- then FOLTL(F) is decidable [HWZ00]

Decidable fragments

Are there decidable fragments of FO?

- monadic fragment
- Bernays-Schönfinkel-Ramsey fragment ($\exists^*\forall^* / =$)
- Ackermann fragment ($\exists^*\forall\exists^*$)
- Gödel-Kalmár-Schütte fragment ($\exists^*\forall\forall\exists^*$)
- Skolem fragment
- two-variable fragment
- guarded fragment
- separated fragment
- combinations of the above in multi-sorted signatures
- many description logics are decidable FO fragments in disguise

See [Voi19] for a detailed survey.

THE PAST:
LTL MODULO THEORIES

FOLTL in practice

There have been many developments in the world of first-order theorem proving.

- many good solvers (SPASS, Vampire, E, *etc.*);
- not specific to decidable fragments, however.

Can we use these as a foundation of FOLTL reasoning? Yes.

- See *e.g.*, TSPASS, a solver for monodic first-order temporal logic

FOLTL in practice

However, the **monodic** temporal fragment is limited for applications.

Imagine modeling an **update** query on an SQL database:

update R **set** x = x - 1 **where** x > 0

In FOLTL, it can become something like:

$$\forall k \forall x \left(\begin{array}{l} \quad \big(R(k,x) \wedge x > 0 \rightarrow \mathsf{X}(\neg R(k,x) \wedge R(k,x-1))\big) \\ \wedge \big(R(k,x) \wedge x \leqslant 0 \rightarrow \mathsf{X} R(k,x)\big) \end{array} \right)$$

We need to accept dealing with **semi-decidability**.

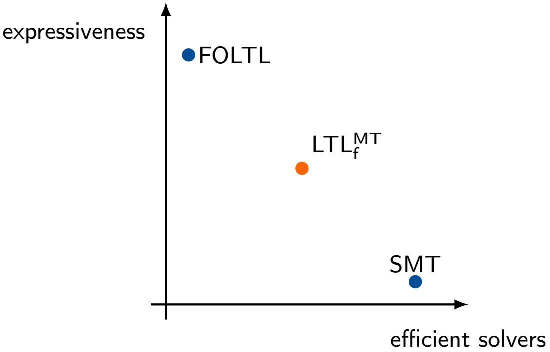
Satisfiability Modulo Theories

There is another approach at **first-order** reasoning:

- satisfiability modulo theories
- **bottom-up** approach:
  - let's add to SAT solvers bits of tractable first-order reasoning step by step
- SMT solvers are strong at quantifier-free reasoning over specific theories useful for program verification
  - linear integer arithmetics/real arithmetics
  - arrays
  - bitvectors
  - constrained Horn clauses
  - algebraic data types
- sometimes quantification works, as well

LTL_f modulo theories

LTL_f^{MT} is a sweet spot in the specification of infinite-state properties. [IJCAI 22]



$\mathsf{LTL_f}$ modulo theories

LTL_f^{MT} extends $\mathsf{LTL_f}$ by introducing **some** first-order elements.

- let Σ be a first-order signature and T a Σ-theory
- LTL_f^{MT} is interpreted over finite words of T-structures
- predicates and function symbols are **rigid**
- constants are **non-rigid**, *i.e.*, change over time
  - this is **not** covered by existing decidability results about FOLTL

$\mathsf{LTL_f}$ modulo theories

$\mathsf{LTL_f^{MT}}$ extends $\mathsf{LTL_f}$ by introducing **some** first-order elements.

- terms can refer to the value of constants at the **next** or **previous** state.

$$t \coloneqq c \mid x \mid f(t_1, \ldots, t_n) \mid \bigcirc c \mid \circledsim c \mid \lhd c \mid \tilde{\lhd} c$$

- $\mathsf{LTL_f}$ propositions are replaced by first-order sentences

$$\lambda \coloneqq p(t_1, \ldots, t_n) \mid \neg\lambda \mid \lambda \vee \lambda \mid \lambda \wedge \lambda \mid \exists x\, \lambda \mid \forall x\, \lambda$$

$$\phi \coloneqq \lambda \mid \neg\phi \mid \phi \vee \phi \mid \phi \wedge \phi \mid \mathsf{X}\phi \mid \widetilde{\mathsf{X}}\phi \mid \mathsf{Y}\phi \mid \mathsf{Z}\phi$$
$$\mid \phi\, \mathsf{U}\, \phi \mid \phi\, \mathsf{R}\, \phi \mid \phi\, \mathsf{S}\, \phi \mid \phi\, \mathsf{T}\, \phi$$

$\mathsf{LTL_f}$ modulo theories

$\mathsf{LTL_f^{MT}}$ extends $\mathsf{LTL_f}$ by introducing **some** first-order elements.

- terms can refer to the value of constants at the **next** or **previous** state.

$$t \coloneqq c \mid x \mid f(t_1, \ldots, t_n) \mid \bigcirc c \mid \widetilde{\bigcirc}\, c \mid \lhd c \mid \widetilde{\lhd} c$$

- $\mathsf{LTL_f}$ propositions are replaced by first-order sentences

$$\begin{aligned}
\lambda &\coloneqq p(t_1, \ldots, t_n) \mid \neg\lambda \mid \lambda \lor \lambda \mid \lambda \land \lambda \mid \exists x\, \lambda \mid \forall x\, \lambda \\
\phi &\coloneqq \lambda \mid \neg\phi \mid \phi \lor \phi \mid \phi \land \phi \mid \mathsf{X}\phi \mid \widetilde{\mathsf{X}}\phi \mid \mathsf{Y}\phi \mid \mathsf{Z}\phi \\
&\quad \mid \phi\, \mathsf{U}\, \phi \mid \phi\, \mathsf{R}\, \phi \mid \phi\, \mathsf{S}\, \phi \mid \phi\, \mathsf{T}\, \phi
\end{aligned}$$

$\mathsf{LTL_f}$ modulo theories

$\mathsf{LTL_f^{MT}}$ extends $\mathsf{LTL_f}$ by introducing **some** first-order elements.

- terms can refer to the value of constants at the **next** or **previous** state.

$$t \coloneqq c \mid x \mid f(t_1, \ldots, t_n) \mid \bigcirc c \mid \widetilde{\bigcirc} c \mid \triangleleft c \mid \widetilde{\triangleleft} c$$

- $\mathsf{LTL_f}$ propositions are replaced by first-order sentences

$$\lambda \coloneqq p(t_1, \ldots, t_n) \mid \neg\lambda \mid \lambda \vee \lambda \mid \lambda \wedge \lambda \mid \exists x\, \lambda \mid \forall x\, \lambda$$

$$\phi \coloneqq \lambda \mid \neg\phi \mid \phi \vee \phi \mid \phi \wedge \phi \mid \mathsf{X}\phi \mid \widetilde{\mathsf{X}}\phi \mid \mathsf{Y}\phi \mid \mathsf{Z}\phi$$

$$\mid \phi\ \mathsf{U}\ \phi \mid \phi\ \mathsf{R}\ \phi \mid \phi\ \mathsf{S}\ \phi \mid \phi\ \mathsf{T}\ \phi$$

$\mathsf{LTL_f}$ modulo theories

LTL_f^{MT} extends $\mathsf{LTL_f}$ by introducing **some** first-order elements.

- terms can refer to the value of constants at the **next** or **previous** state.

$$t \coloneqq c \mid x \mid f(t_1, \ldots, t_n) \mid \bigcirc c \mid \widetilde{\bigcirc} c \mid \lhd c \mid \widetilde{\lhd} c$$

- $\mathsf{LTL_f}$ propositions are replaced by first-order sentences

$$\begin{aligned}
\lambda &\coloneqq p(t_1, \ldots, t_n) \mid \neg\lambda \mid \lambda \vee \lambda \mid \lambda \wedge \lambda \mid \exists x\, \lambda \mid \forall x\, \lambda \\
\phi &\coloneqq \lambda \mid \neg\phi \mid \phi \vee \phi \mid \phi \wedge \phi \mid \mathsf{X}\phi \mid \widetilde{\mathsf{X}}\phi \mid \mathsf{Y}\phi \mid \mathsf{Z}\phi \\
&\quad \mid \phi \,\mathsf{U}\, \phi \mid \phi \,\mathsf{R}\, \phi \mid \phi \,\mathsf{S}\, \phi \mid \phi \,\mathsf{T}\, \phi
\end{aligned}$$

Examples

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

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

Examples

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

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

Examples

$$\mathrm{G}(a = 2b) \qquad (a < b)\ \mathrm{U}\ (b = 0) \qquad \mathrm{G}(a > 5) \wedge \mathrm{F}(a = 0)$$

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

Examples

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

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

Examples

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

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

Examples

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

$$b = 1 \wedge \mathsf{G}(\circledtilde\, b = b + 1 \wedge a = 2b)$$

$$\mathsf{G}(p(a) \rightarrow \widetilde{\mathsf{X}} p(\lhd\, a + 1))$$

Examples

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

$$b = 1 \land \mathsf{G}(\circledsim\, b = b + 1 \land a = 2b)$$

$$\mathsf{G}(p(a) \rightarrow \widetilde{\mathsf{X}} p(\lhd\, a + 1))$$

Examples

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

$$b = 1 \wedge \mathsf{G}(\circledtilde\, b = b + 1 \wedge a = 2b)$$

$$\mathsf{G}(p(a) \to \widetilde{\mathsf{X}} p(\lhd\, a + 1))$$

Examples

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

$$b = 1 \land \mathsf{G}(\circleddash\!\!\!\sim\, b = b + 1 \land a = 2b)$$

$$\mathsf{G}(p(a) \rightarrow \widetilde{\mathsf{X}} p(\lhd\, a + 1))$$

(Un)decidability

$\mathsf{LTL}_f^{\mathsf{MT}}$ is **undecidable**, but:

- it is **semi-decidable**
- so we can get something useful, sometimes:
  - models of satisfiable formulas
  - counterexamples of specifications over bugged systems
- moreover, we have some interesting **decidable** fragments [ECAI 23]
- why **finite traces**?
  - on **infinite traces**, $\mathsf{LTL}^{\mathsf{MT}}$ is not even semi-decidable.

Decidable fragments

(MC) Formulas over LRA where all **iteration conditions** are **monotonicity constraints**, *i.e.*, variable-to-variable or variable-to-constant comparisons, *e.g.*:

$$a < 0 \wedge b = 1 \wedge ((\bigcirc b > b \wedge \bigcirc a \leqslant a) \,\mathsf{U}\, a = b)$$

(IPC) Formulas over LIA where all **iteration conditions** are **integer periodicity constraints**, *e.g.*:

$$(b \equiv_3 a) \,\mathsf{U}\, (a > 42) \wedge \mathsf{F}(a + b = c)$$

Iteration conditions:

- α in $\alpha \,\mathsf{U}\, \beta$
- β in $\alpha \,\mathsf{R}\, \beta$

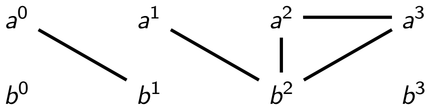
Decidable fragments

(BL) **Bounded lookback** formulas, that generalize the above two by requiring that cross-state interaction is restricted to finitely many steps back. What does it mean?

Decidable fragments

Bounded lookback formula

$$p(a, \bigcirc b) \,\mathsf{U}\, (\bigcirc a = a + b)$$



$p(a^0, b^1)$ $\quad$ $p(a^1, b^2)$ $\quad$ $a^3 = a^2 + b^2$

Bounded lookback formula

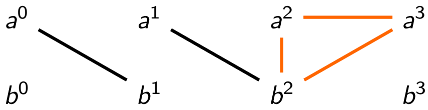
$$p(a, \bigcirc b) \,\mathsf{U}\, (\bigcirc a = a + b)$$

$p(a^0, b^1)$ $p(a^1, b^2)$ $a^3 = a^2 + b^2$

Decidable fragments

Bounded lookback formula

$$p(a, \bigcirc b) \,\mathsf{U}\, (\bigcirc a = a + b)$$



$p(a^0, b^1)$ $\quad$ $p(a^1, b^2)$ $\quad$ $a^3 = a^2 + b^2$

Bounded lookback formula

$$p(a, \bigcirc b) \; \mathrm{U} \; (\bigcirc a = a + b)$$

$p(a^0, b^1)$ $\quad p(a^1, b^2)$ $\quad a^3 = a^2 + b^2$

Decidable fragments

Non-BL formula

$$a < 0 \wedge b = 1 \wedge ((\bigcirc\, b > b \wedge \bigcirc\, a \leqslant a) \,\mathsf{U}\, a = b)$$

$$\begin{tikzcd}
a^0 \arrow[r, dash] & a^1 \arrow[r, dash] & a^2 \arrow[r, dash] & a^3 \arrow[d, dash] \\
b^0 \arrow[r, dash] & b^1 \arrow[r, dash] & b^2 \arrow[r, dash] & b^3
\end{tikzcd}$$

$$\begin{array}{cccc}
a^1 \leqslant a^0 & a^2 \leqslant a^1 & a^3 \leqslant a^2 & a^3 = b^3 \\
b^1 > b^0 & b^2 > b^1 & b^3 > b^2 &
\end{array}$$

Decidable fragments

Non-BL formula

$$a < 0 \wedge b = 1 \wedge ((\bigcirc b > b \wedge \bigcirc a \leqslant a) \,\mathsf{U}\, a = b)$$

$$\begin{tikzcd}
a^0 \arrow[r, dash] & a^1 \arrow[r, dash] & a^2 \arrow[r, dash] & a^3 \arrow[d, dash] \\
b^0 \arrow[r, dash] & b^1 \arrow[r, dash] & b^2 \arrow[r, dash] & b^3
\end{tikzcd}$$

$$\begin{array}{cccc}
a^1 \leqslant a^0 & a^2 \leqslant a^1 & a^3 \leqslant a^2 & a^3 = b^3 \\
b^1 > b^0 & b^2 > b^1 & b^3 > b^2 &
\end{array}$$

Non-BL formula

$$a < 0 \wedge b = 1 \wedge ((\bigcirc\, b > b \wedge \bigcirc\, a \leqslant a) \,\mathsf{U}\, a = b)$$

$$\begin{tikzcd}
a^0 \arrow[r, dash] & a^1 \arrow[r, dash] & a^2 \arrow[r, dash] & a^3 \arrow[d, dash] \\
b^0 \arrow[r, dash] & b^1 \arrow[r, dash] & b^2 \arrow[r, dash] & b^3
\end{tikzcd}$$

$$\begin{array}{cccc}
a^1 \leqslant a^0 & a^2 \leqslant a^1 & a^3 \leqslant a^2 & a^3 = b^3 \\
b^1 > b^0 & b^2 > b^1 & b^3 > b^2 &
\end{array}$$

Decidable fragments

Non-BL formula

$$a < 0 \wedge b = 1 \wedge ((\bigcirc b > b \wedge \bigcirc a \leqslant a) \,\mathsf{U}\, a = b)$$

$$\begin{tikzcd}
a^0 \arrow[r, dash] & a^1 \arrow[r, dash] & a^2 \arrow[r, dash] & a^3 \arrow[d, dash] \\
b^0 \arrow[r, dash] & b^1 \arrow[r, dash] & b^2 \arrow[r, dash] & b^3
\end{tikzcd}$$

$$\begin{array}{cccc}
a^1 \leqslant a^0 & a^2 \leqslant a^1 & a^3 \leqslant a^2 & a^3 = b^3 \\
b^1 > b^0 & b^2 > b^1 & b^3 > b^2 &
\end{array}$$

Decidable fragments

Non-BL formula

$$a < 0 \wedge b = 1 \wedge ((\bigcirc b > b \wedge \bigcirc a \leqslant a) \,\mathsf{U}\, a = b)$$

$$\begin{tikzcd}
a^0 \arrow[r, dash] & a^1 \arrow[r, dash] & a^2 \arrow[r, dash] & a^3 \arrow[d, dash] \\
b^0 \arrow[r, dash] & b^1 \arrow[r, dash] & b^2 \arrow[r, dash] & b^3
\end{tikzcd}$$

$$\begin{array}{cccc}
a^1 \leqslant a^0 & a^2 \leqslant a^1 & a^3 \leqslant a^2 & a^3 = b^3 \\
b^1 > b^0 & b^2 > b^1 & b^3 > b^2 &
\end{array}$$

Decidable fragments

(NCS) Formulae without **cross-state comparisons**, *i.e.*, without any $\bigcirc c$, $\circledsim c$, *e.g.*:

$$(a > b \,\mathsf{U}\, a + b = 2c) \wedge \mathsf{G}(a + b > 0)$$

(FX) Formulas where the only temporal operators are F, X, and $\widetilde{\mathsf{X}}$, *e.g.*:

$$\mathsf{F}(p(\bigcirc a) \wedge \mathsf{X}(\neg p(a))) \wedge \mathsf{XF}(r(a, b) \vee r(\bigcirc a, b))$$

The SAT encoding of Reynolds' tableau can be extended to an **SMT encoding** for $\mathrm{LTL}_\mathrm{f}^\mathrm{MT}$.

- implemented in our BLACK solver[1] for arithmetic theories;
- given to **off-the-shelf** SMT solvers such as Z3 [MB08], cvc5 [Bar+22], *etc.*

[1] `https://www.black-sat.org`

Verification of $\mathrm{LTL}_f^{\mathrm{MT}}$ properties

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

Knowledge-base driven Dynamic Systems (KDS):

- infinite-state transition systems

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

- states are structures over the **first-order theory** K
- $I(X)$, $T(X, X')$, $F(X)$ 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 $\mathrm{LTL}_f^{\mathrm{MT}}$ formula:

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

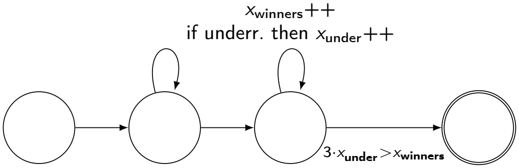
$$\gamma \equiv \psi_D \wedge \neg\phi$$

- 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
- this needs **non-rigid predicates**: only semi-decidable!

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



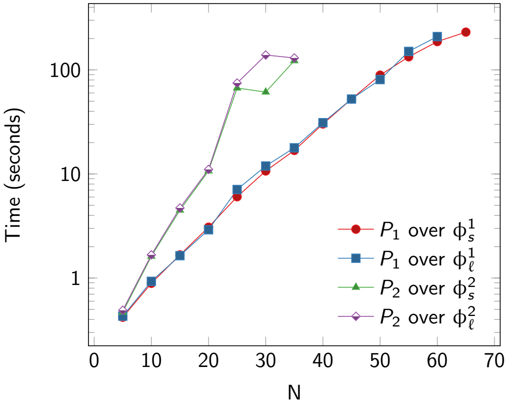
$$\phi_s^1 \equiv \mathsf{G}(x_{state} = final \rightarrow 2x_{\text{under}} > x_{\text{winners}})$$

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

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



THE PRESENT:
FIRST-ORDER AUTOMATA

Going beyond $\mathsf{LTL}_f^{\mathsf{MT}}$

$\mathsf{LTL}_f^{\mathsf{MT}}$ is limited in many ways:

- many complex systems need **evolving predicates**;
- **nesting** of quantifiers and temporal operators is often essential
- we want to approach **full** FOLTL.

The need for automata

In the finite-state setting, automata are essential:

- operational counterpart of temporal logics
- basis of many algorithms and techniques

We want a class of automata for FOLTL.

First-order automata

What may an infinite-state first-order automaton look like?

- **states**: first-order structures over a state signature Γ;
- **letters**: first-order structures over an alphabet signature Σ;
- **initial states**: a class of Γ-structures;
- **final states**: a class of Γ-structures;
- **transition**: a relation between:
  - a Γ-structure (source state),
  - a Σ-structure (letter), and
  - another Γ-structure (dest. state).

But this is algorithmically untractable.

Symbolic representation

A **finite-state** automaton:

$$A = \langle \Sigma, Q, Q_0, \Delta, F \rangle$$

A **symbolic** finite-state automaton:

$$\mathcal{A} = \langle \Sigma, X, I(X), T(X, \Sigma, X'), F(X) \rangle$$

Symbolic representation

A **finite-state** automaton:

$$A = \langle \Sigma, Q, Q_0, \Delta, F \rangle$$

A **symbolic** finite-state automaton:

$$\mathcal{A} = \langle \Sigma, X, I(X), T(X, \Sigma, X'), F(X) \rangle$$

A **first-order** automaton:

$$\mathrm{A} = \langle \Sigma, \Gamma, \phi_I, \phi_T, \phi_F \rangle$$

where:

- Σ is the **word** signature and Γ is the **state** signature;
- ϕ_I and ϕ_F are Γ-sentences;
- ϕ_T is a sentence over $\Gamma \cup \Sigma \cup \Gamma'$.

First-order automata

A **first-order** automaton:

$$\mathcal{A} = \langle \Sigma, \Gamma, \phi_I, \phi_T, \phi_F \rangle$$

where:

- Σ is the **word** signature and Γ is the **state** signature;
- ϕ_I and ϕ_F are Γ-sentences;
- ϕ_T is a sentence over $\Gamma \cup \Sigma \cup \Gamma'$.

A word (sequence of Σ-structures) $\overline{\sigma} = \langle \sigma_0, \dots, \sigma_{n-1} \rangle$ is accepted iff there is a **trace** (sequence of Γ-structures) $\overline{\rho} = \langle \rho_0, \dots, \rho_n \rangle$ such that:

- $\rho_0 \models \phi_0$;
- $\rho_i \cup \sigma_i \cup \rho'_{i+1} \models \phi_T$;
- $\rho_n \models \phi_F$.

First-order T-regular languages

A set of first-order words whose structures come from a theory T is a **T-language**.

A T-language is **first-order T-regular** if there is a first-order automaton accepting it.

First-order T-regular languages

Theorem (Closure properties)

*First-order T-regular languages are **closed** by:*

- *union;*
- *intersection;*
- *concatenation;*
- *Kleene star.*

As an example we see how to prove concatenation.

Concatenation

We have two automata $A_1 = \langle \Sigma, \Gamma_1, \phi_0^1, \phi_T^1, \phi_F^1 \rangle$ and $A_2 = \langle \Sigma, \Gamma_2, \phi_0^2, \phi_T^2, \phi_F^2 \rangle$.

We want $A = \langle \Sigma, \Gamma, \phi_0, \phi_T, \phi_F \rangle$ such that $\mathcal{L}(A) = \mathcal{L}(A_1) \cdot \mathcal{L}(A_2)$.

- suppose for a moment that ϕ_T can be an **existential second-order** sentence;
- then, what about:

$$\begin{aligned} \phi_T \equiv (p \wedge \phi_T^1 \wedge p') \vee (\neg p \wedge \phi_T^2 \wedge \neg p') \vee \\ \Big(p \wedge \neg p' \wedge \phi_F^1 \wedge \exists \Gamma'' \big(\phi_0^2[\Gamma/\Gamma''] \wedge \phi_T^2[\Gamma/\Gamma'']\big)\Big) \end{aligned}$$

with $\Gamma = \Gamma_1 \cup \Gamma_2 \cup \{p\}$.

Concatenation

What about these existential second-order quantifiers?

$$\phi_T \equiv (p \wedge \phi_T^1 \wedge p') \vee (\neg p \wedge \phi_T^2 \wedge \neg p') \vee \\ \Big(p \wedge \neg p' \wedge \phi_F^1 \wedge \exists \Gamma'' \big(\phi_0^2[\Gamma/\Gamma''] \wedge \phi_T^2[\Gamma/\Gamma'']\big)\Big)$$

Concatenation

What about these existential second-order quantifiers?

$$\phi_T \equiv \exists \Gamma'' \begin{bmatrix} (p \wedge \phi_T^1 \wedge p') \vee (\neg p \wedge \phi_T^2 \wedge \neg p') \vee \\ \left(p \wedge \neg p' \wedge \phi_F^1 \wedge (\phi_0^2[\Gamma/\Gamma''] \wedge \phi_T^2[\Gamma/\Gamma''])\right) \end{bmatrix}$$

Concatenation

What about these existential second-order quantifiers?

$$\phi_T \equiv \exists \Gamma'' \begin{bmatrix} (p \wedge \phi^1_T \wedge p') \vee (\neg p \wedge \phi^2_T \wedge \neg p') \vee \\ \left(p \wedge \neg p' \wedge \phi^1_F \wedge (\phi^2_0[\Gamma/\Gamma''] \wedge \phi^2_T[\Gamma/\Gamma''])\right) \end{bmatrix}$$

Now the predicates in Γ'' can be added to Γ itself:

- the semantics of the automata will require the existence of their interpretation;
- we get back a first-order automaton on an extended state signature.

Concatenation

What about these existential second-order quantifiers?

$$\phi_T \equiv \begin{bmatrix} (p \wedge \phi_T^1 \wedge p') \vee (\neg p \wedge \phi_T^2 \wedge \neg p') \vee \\ \left(p \wedge \neg p' \wedge \phi_F^1 \wedge (\phi_0^2[\Gamma/\Gamma''] \wedge \phi_T^2[\Gamma/\Gamma''])\right) \end{bmatrix}$$

Now the predicates in Γ'' can be added to Γ itself:

- the semantics of the automata will require the existence of their interpretation;
- **we get back a first-order automaton on an extended state signature.**

Complementation

Complementation is missing from the picture:

- easy for **deterministic** automata;
- so **determinization** is the key;
- how to symbolically represent the subset construction of a first-order automaton?
  - we are working on it. . .

Capturing FOLTL

A major feature of automata is that they capture temporal logic.

Can we capture FOLTL with first-order automata?

Capturing FOLTL

We need some ingredients.

Definition (Stepped normal form)

Given an FOLTL formula ϕ, the *step normal form* of ϕ, denoted $\mathsf{snf}(\phi)$, is the formula defined recursively as follows:

- $\mathsf{snf}(p(t_1, \ldots, t_n)) = p(t_1, \ldots, t_n)$ and $\mathsf{snf}(t_1 = t_2) = (t_1 = t_2)$;
- $\mathsf{snf}(Qx\phi) = Qx\,\mathsf{snf}(\phi)$, where $Q \in \{\forall, \exists\}$ and x is a first-order variable;
- $\mathsf{snf}(\neg\phi) = \neg\,\mathsf{snf}(\phi)$;
- $\mathsf{snf}(\phi_1 \circ \phi_2) = \mathsf{snf}(\phi_1) \circ \mathsf{snf}(\phi_2)$, where $\circ \in \{\wedge, \vee\}$
- $\mathsf{snf}(\circ\phi) = \circ\phi$ where $\circ \in \{\mathsf{X}, \mathsf{Y}, \widetilde{\mathsf{X}}, \mathsf{Z}\}$;
- $\mathsf{snf}(\phi_1 \mathbin{\mathsf{U}} \phi_2) = \mathsf{snf}(\phi_2) \vee (\mathsf{snf}(\phi_1) \wedge \mathsf{X}(\phi_1 \mathbin{\mathsf{U}} \phi_2))$;
- $\mathsf{snf}(\phi_1 \mathbin{\mathsf{R}} \phi_2) = \mathsf{snf}(\phi_2) \wedge (\mathsf{snf}(\phi_1) \vee \widetilde{\mathsf{X}}(\phi_1 \mathbin{\mathsf{R}} \phi_2))$.

Definition (Closure)

The *closure* of a FOLTL sentence ϕ is the set $C(\phi)$ defined as follows:

- $X\phi \in C(\phi)$;
- $\psi \in C(\phi)$ for any subformula ψ of ϕ (including itself);
- for any $\phi_1 \mathbin{U} \phi_2 \in C(\phi)$ we have $X(\phi_1 \mathbin{U} \phi_2) \in C(\phi)$;
- for any $\phi_1 \mathbin{R} \phi_2 \in C(\phi)$ we have $\widetilde{X}(\phi_1 \mathbin{R} \phi_2) \in C(\phi)$.

Capturing FOLTL

We also define the following subsets of the closure:

$$\mathsf{XS} = \{\mathsf{xs}_\psi \mid \mathsf{X}\psi \in \mathsf{C}(\phi)\}$$
$$\widetilde{\mathsf{X}}\mathsf{S} = \{\mathsf{ws}_\psi \mid \widetilde{\mathsf{X}}\psi \in \mathsf{C}(\phi)\}$$

where xs_ψ and ws_ψ are predicates of the arity n corresponding to the number of free first-order variables in ψ.

Capturing FOLTL

Let ϕ be a FOLTL sentence. The automaton of ϕ is $\mathcal{A}(\phi) = \langle \Sigma, \Gamma, \phi_0, \phi_T, \phi_F \rangle$ where:

- the state signature is $\Gamma = \mathsf{XS} \cup \widetilde{\mathsf{X}}\mathsf{S}$;
- the initial and final conditions are:

$$\phi_0 \equiv \mathsf{xs}_\phi$$

$$\phi_F \equiv \bigwedge_{\mathsf{ws}_\psi \in \widetilde{\mathsf{X}}\mathsf{S}} \forall \overline{x}.\mathsf{ws}_\psi(\overline{x}) \wedge \bigwedge_{\mathsf{xs}_\psi \in \mathsf{XS}} \forall \overline{x}.\neg \mathsf{xs}_\psi(\overline{x})$$

- the transition relation is:

$$\phi_T \equiv \bigwedge_{s_\psi \in \mathsf{XS} \cup \widetilde{\mathsf{X}}\mathsf{S}} \forall \overline{x}.[s_\psi(\overline{x}) \leftrightarrow \mathsf{snf}'_S(\psi(\overline{x}))]$$

Checking (non-)emptiness

Of course checking emptiness of first-order automata is **undecidable**.

However, we can state a semi-algorithm for non-emptiness with some assumptions.

Checking (non-)emptiness

For $k > 0$, let:

$$[\![A]\!]_k^F \equiv \phi_0^0 \wedge \bigwedge_{i=0}^{k-1} \phi_T^k \wedge \phi_F^k$$

```
procedure NonEmpty(A)
    k ← 0
    while true do
        if [[A]]_k^F is satisfiable then
            return not empty
        end if
        k ← k + 1
    end while
end procedure
```

Decidable fragments

Recall the result from the literature we cited before.

For a fragment F such that:

- the underlying FO fragment is decidable; and
- temporal formulas are **monodic**;
  - only one **free variable** inside temporal operators;
- constants in the signature are considered **rigid**;
- then FOLTL(F) is decidable [HWZ00]

Decidable fragments

Can we recover this result from first-order automata? **Yes.**

- if we follow the encoding we notice that **monodic** FOLTL sentences translate into automata with **monadic** state signature Γ;
- monadic predicates are easier to deal with (*e.g.*, monadic FO is decidable);
- we can translate a monadic Γ into a Γ containing only **propositions**;
- emptyness for this kind of automata is decidable (they are almost finite-state).

CONCLUSIONS

First-order automata in practice

Our satisfiability checking tool BLACK is currently being refactored to support full FOLTL through first-order automata.

- experimental results soon, maybe another iFM^2 talk?

Better algorithms

Can we go beyond the simple unraveling semi-algorithm shown before?

For better **practical** results we need more **theory**.

Interesting paths we are exploring:

- fixpoint computation based on **second-order quantifier elimination** [GSS08];
- encoding into **constrained Horn clauses** [GB19];

What we have not seen today

We can add **past operators** to FOLTL.

- if we start from **pure-past** FOLTL sentences, we directly obtain **deterministic** first-order automata;
- this happens with pure-past LTL as well and is crucial for **reactive synthesis**;
- 2EXPTIME for LTL, EXPTIME for pure-past LTL, because of this fact;
- can this help with **reactive synthesis** for FOLTL specifications?

Conclusions

We want first-order temporal logics to become practical tools for infinite-state specification and verification.

- long way ahead;
- little steps already done;
- first-order automata as a reasoning and algorithmic tool;
- many theoretical and algorithmic developments missing yet.

Conclusions

We want first-order temporal logics to become practical tools for infinite-state specification and verification.

- long way ahead;
- little steps already done;
- first-order automata as a reasoning and algorithmic tool;
- many theoretical and algorithmic developments missing yet.

Thank you!

Questions?

REFERENCES

References

[AD94] Rajeev Alur and David L. Dill. 'A Theory of Timed Automata'. In: *Theor. Comput. Sci.* 126.2 (1994), pp. 183–235. doi: 10.1016/0304-3975(94)90010-8.

[AH94] Rajeev Alur and Thomas A. Henzinger. 'A Really Temporal Logic'. In: *Journal of the ACM* 41.1 (1994), pp. 181–204. doi: 10.1145/174644.174651.

[Alu+05] Rajeev Alur, Michael Benedikt, Kousha Etessami, Patrice Godefroid, Thomas W. Reps and Mihalis Yannakakis. 'Analysis of recursive state machines'. In: *ACM Trans. Program. Lang. Syst.* 27.4 (2005), pp. 786–818. doi: 10.1145/1075382.1075387.

[Alu+92] Rajeev Alur, Costas Courcoubetis, Thomas A. Henzinger and Pei-Hsin Ho. 'Hybrid Automata: An Algorithmic Approach to the Specification and Verification of Hybrid Systems'. In: *Hybrid Systems*. Ed. by Robert L. Grossman, Anil Nerode, Anders P. Ravn and Hans Rischel. Vol. 736. Lecture Notes in Computer Science. Springer, 1992, pp. 209–229. doi: 10.1007/3-540-57318-6_30.

References

[AM04] Rajeev Alur and P. Madhusudan. 'Visibly pushdown languages'. In: *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*. Ed. by László Babai. ACM, 2004, pp. 202–211. doi: 10.1145/1007352.1007390.

[Bar+22] Haniel Barbosa, Clark W. Barrett, Martin Brain, Gereon Kremer, Hanna Lachnitt, Makai Mann, Abdalrhman Mohamed, Mudathir Mohamed, Aina Niemetz, Andres Nötzli, Alex Ozdemir, Mathias Preiner, Andrew Reynolds, Ying Sheng, Cesare Tinelli and Yoni Zohar. 'cvc5: A Versatile and Industrial-Strength SMT Solver'. In: *Proceedings of the 28th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Ed. by Dana Fisman and Grigore Rosu. Vol. 13243. Lecture Notes in Computer Science. Springer, 2022, pp. 415–442. doi: 10.1007/978-3-030-99524-9_24.

References

[BK18] James Brotherston and Max I. Kanovich. 'On the Complexity of Pointer Arithmetic in Separation Logic'. In: *Proceedings of the 16th Asian Symposium on Programming Languages and Systems*. Ed. by Sukyoung Ryu. Vol. 11275. Lecture Notes in Computer Science. Springer, 2018, pp. 329–349. doi: 10.1007/978-3-030-02768-1_18.

[BMP10] Massimo Benerecetti, Stefano Minopoli and Adriano Peron. 'Analysis of Timed Recursive State Machines'. In: *Proceedings of the 17th International Symposium on Temporal Representation and Reasoning*. Ed. by Nicolas Markey and Jef Wijsen. IEEE Computer Society, 2010, pp. 61–68. doi: 10.1109/TIME.2010.10.

[Boj+06] Mikolaj Bojanczyk, Anca Muscholl, Thomas Schwentick, Luc Segoufin and Claire David. 'Two-Variable Logic on Words with Data'. In: *21th IEEE Symposium on Logic in Computer Science (LICS 2006), 12-15 August 2006, Seattle, WA, USA, Proceedings*. IEEE Computer Society, 2006, pp. 7–16. doi: 10.1109/LICS.2006.51.

[BZ83] Daniel Brand and Pitro Zafiropulo. 'On Communicating Finite-State Machines'. In: *J. ACM* 30.2 (1983), pp. 323–342. doi: 10.1145/322374.322380.

References

[CGM13] Diego Calvanese, Giuseppe De Giacomo and Marco Montali. 'Foundations of data-aware process analysis: a database theory perspective'. In: *Proceedings of the 32nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*. Ed. by Richard Hull and Wenfei Fan. ACM, 2013, pp. 1–12. doi: 10.1145/2463664.2467796.

[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. url: https://doi.org/10.1016/j.ic.2019.104502.

[CMP22] Michele Chiari, Dino Mandrioli and Matteo Pradella. 'A First-Order Complete Temporal Logic for Structured Context-Free Languages'. In: *Log. Methods Comput. Sci.* 18.3 (2022). doi: 10.46298/lmcs-18(3:11)2022.

References

[CYO01] Cristiano Calcagno, Hongseok Yang and Peter W. O'Hearn. 'Computability and Complexity Results for a Spatial Assertion Language for Data Structures'. In: *Proceedings of the 21st Conference on Foundations of Software Technology and Theoretical Computer Science*. Ed. by Ramesh Hariharan, Madhavan Mukund and V. Vinay. Vol. 2245. Lecture Notes in Computer Science. Springer, 2001, pp. 108–119. doi: 10.1007/3-540-45294-X_10.

[DLN07] Stéphane Demri, Ranko Lazic and David Nowak. 'On the freeze quantifier in Constraint LTL: Decidability and complexity'. In: *Inf. Comput.* 205.1 (2007), pp. 2–24. doi: 10.1016/j.ic.2006.08.003.

[DLV19] Alin Deutsch, Yuliang Li and Victor Vianu. 'Verification of Hierarchical Artifact Systems'. In: *ACM Trans. Database Syst.* 44.3 (2019), 12:1–12:68. doi: 10.1145/3321487.

[DQ21] Stéphane Demri and Karin Quaas. 'Concrete domains in logics: a survey'. In: *ACM SIGLOG News* 8.3 (2021), pp. 6–29. doi: 10.1145/3477986.3477988. url: https://doi.org/10.1145/3477986.3477988.

References

[Dro+17] Manfred Droste, Stefan Dück, Dino Mandrioli and Matteo Pradella. 'Weighted Operator Precedence Languages'. In: *Procedings of the 42nd International Symposium on Mathematical Foundations of Computer Science*. Ed. by Kim G. Larsen, Hans L. Bodlaender and Jean-François Raskin. Vol. 83. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017, 31:1–31:15. doi: `10.4230/LIPIcs.MFCS.2017.31`.

[DV13] Giuseppe De Giacomo and Moshe Y. Vardi. 'Linear Temporal Logic and Linear Dynamic Logic on Finite Traces'. In: *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*. Ed. by Francesca Rossi. IJCAI/AAAI, 2013, pp. 854–860.

[ECAI 23] Luca Geatti, Alessandro Gianola, Nicola Gigante and Sarah Winkler. 'Decidable Fragments of LTLf Modulo Theories'. In: *Proceedings of the 26th European Conference on Artificial Intelligence*. 2023.

References

[Fel+23] Paolo Felli, Marco Montali, Fabio Patrizi and Sarah Winkler. 'Monitoring Arithmetic Temporal Properties on Finite Traces'. In: *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023*. Ed. by Brian Williams, Yiling Chen and Jennifer Neville. AAAI Press, 2023, pp. 6346–6354. doi: 10.1609/aaai.v37i5.25781.

[GB19] 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*. IEEE, 2019, pp. 6–10. doi: 10.1109/SYNASC49474.2019.00010.

[Ghi+23] Silvio Ghilardi, Alessandro Gianola, Marco Montali and Andrey Rivkin. 'Safety Verification and Universal Invariants for Relational Action Bases'. In: *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*. ijcai.org, 2023, pp. 3248–3257. doi: 10.24963/ijcai.2023/362.

[GSS08] Dov M. Gabbay, Renate A. Schmidt and Andrzej Szalas. *Second-Order Quantifier Elimination - Foundations, Computational Aspects and Applications*. Vol. 12. Studies in logic : Mathematical logic and foundations. College Publications, 2008. isbn: 978-1-904987-56-7.

[HWZ00] Ian M. Hodkinson, Frank Wolter and Michael Zakharyaschev. 'Decidable fragment of first-order temporal logics'. In: *Ann. Pure Appl. Log.* 106.1-3 (2000), pp. 85–134. doi: 10.1016/S0168-0072(00)00018-X.

[IJCAI 22] 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*. 2022, pp. 2641–2647. doi: 10.24963/ijcai.2022/366.

[IX19] Radu Iosif and Xiao Xu. 'Alternating Automata Modulo First Order Theories'. In: *Computer Aided Verification - 31st International Conference, CAV 2019, New York City, NY, USA, July 15-18, 2019, Proceedings, Part II*. Ed. by Isil Dillig and Serdar Tasiran. Vol. 11562. Lecture Notes in Computer Science. Springer, 2019, pp. 43–63. doi: 10.1007/978-3-030-25543-5_3.

[JK09] Kurt Jensen and Lars Michael Kristensen. *Coloured Petri Nets - Modelling and Validation of Concurrent Systems*. Springer, 2009. doi: 10.1007/b95112.

References

[KF94] Michael Kaminski and Nissim Francez. 'Finite-Memory Automata'. In: *Theor. Comput. Sci.* 134.2 (1994), pp. 329–363. doi: 10.1016/0304-3975(94)90242-9.

[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.

[Koy90] Ron Koymans. 'Specifying Real-Time Properties with Metric Temporal Logic'. In: *Real-Time Systems* 2.4 (1990), pp. 255–299. doi: 10.1007/BF01995674.

[LVR22] Jianwen Li, Moshe Y. Vardi and Kristin Y. Rozier. 'Satisfiability checking for Mission-time LTL (MLTL)'. In: *Inf. Comput.* 289.Part (2022), p. 104923. doi: 10.1016/j.ic.2022.104923.

[Man20] Alessio Mansutti. 'Reasoning with separation logics: complexity, expressive power, proof systems'. PhD thesis. University of Paris-Saclay, France, 2020. url: https://tel.archives-ouvertes.fr/tel-03094373.

References

[MB08] Leonardo Mendonça de Moura and Nikolaj S. Bjørner. 'Z3: An Efficient SMT Solver'. In: *Proceedings of the 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Ed. by C. R. Ramakrishnan and Jakob Rehof. Vol. 4963. Lecture Notes in Computer Science. Springer, 2008, pp. 337–340. doi: 10.1007/978-3-540-78800-3_24. url: https://doi.org/10.1007/978-3-540-78800-3%5C_24.

[MN04] Oded Maler and Dejan Nickovic. 'Monitoring Temporal Properties of Continuous Signals'. In: *Proceedings of Formal FORMATS 2004 and FTRTFT 2004*. Ed. by Yassine Lakhnech and Sergio Yovine. Vol. 3253. Lecture Notes in Computer Science. Springer, 2004, pp. 152–166. doi: 10.1007/978-3-540-30206-3_12.

[Mur89] Tadao Murata. 'Petri nets: Properties, analysis and applications'. In: *Proc. IEEE* 77.4 (1989), pp. 541–580. doi: 10.1109/5.24143.

[Seg06] Luc Segoufin. 'Automata and Logics for Words and Trees over an Infinite Alphabet'. In: *Proceedings of the 20th International Workshop on Computer Science Logic*. Ed. by Zoltán Ésik. Vol. 4207. Lecture Notes in Computer Science. Springer, 2006, pp. 41–57. doi: 10.1007/11874683_3.

References

[Voi19] Marco Voigt. 'Decidable fragments of first-order logic and of first-order linear arithmetic with uninterpreted predicates'. PhD thesis. Saarland University, Saarbrücken, Germany, 2019. url: https://publikationen.sulb.uni-saarland.de/handle/20.500.11880/27767.

[Woj99] Arkadiusz Wojna. 'Counter Machines'. In: *Inf. Process. Lett.* 71.5-6 (1999), pp. 193–197. doi: 10.1016/S0020-0190(99)00116-7.