Logics for the specification of hyperproperties

5th November 2024

Jonni Virtema

University of Sheffield, UK

Logics for verification and specification of concurrent systems

Basic setting:

- System (e.g., piece of software or hardware)
  $\rightsquigarrow$ Kripke structure depicting the behaviour of the system
- A single run of the system
  $\rightsquigarrow$ a trace generated by the Kripke structure
- A property of the system (e.g., every request is eventually granted)
  $\rightsquigarrow$ a formula of some formal language expressing the property.

Logics for verification and specification of concurrent systems

Basic setting:

- System (e.g., piece of software or hardware)
  $\rightsquigarrow$ Kripke structure depicting the behaviour of the system
- A single run of the system
  $\rightsquigarrow$ a trace generated by the Kripke structure
- A property of the system (e.g., every request is eventually granted)
  $\rightsquigarrow$ a formula of some formal language expressing the property.

Model checking:

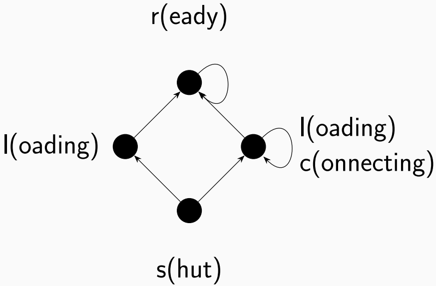
- Check whether a given system satisfies a given specification.

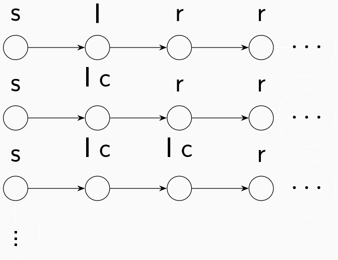
SAT solving:

- Check whether a given specification (or collection of) can be realised.

Traceproperties and hyperproperties

Opening your office computer after holidays:





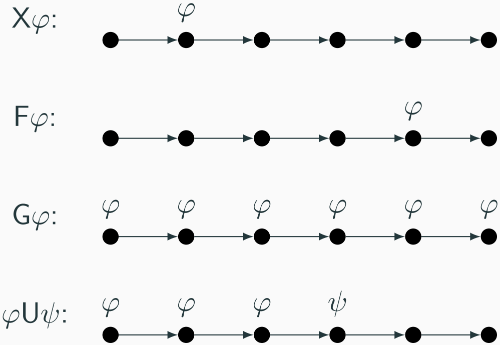
Traceproperties hold in a system if each trace (in isolation) has the property:

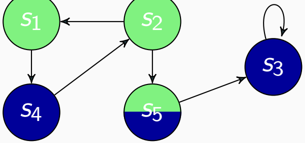
- The computer will be eventually ready (or will be loading forever).

Hyperproperties are properties of sets of traces:

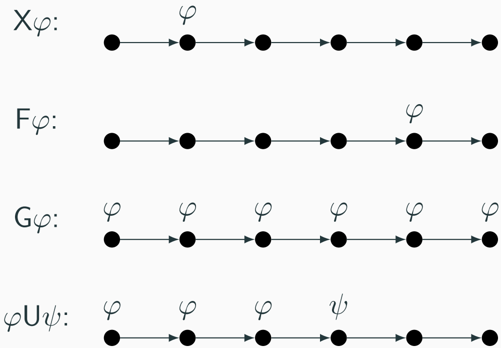
- The computer will be ready in bounded time.

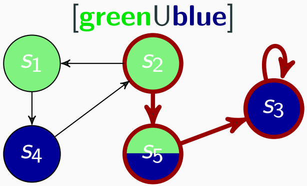
Temporal Logic for traceproperties: Semantics by Example





Temporal Logic for traceproperties: Semantics by Example





Temporal Logic for traceproperties: Semantics by Example

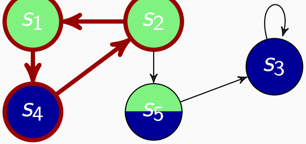
$X\varphi$:

$F\varphi$:

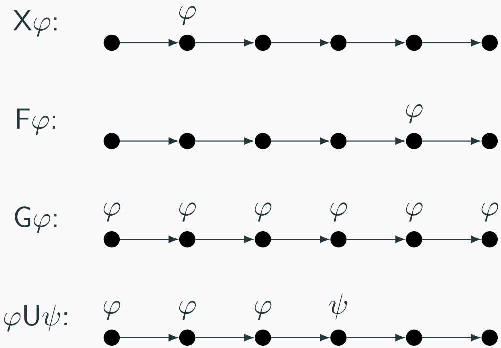
$G\varphi$:

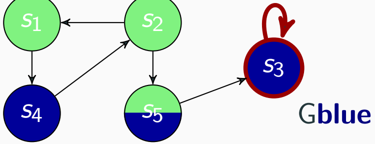
$\varphi U \psi$:

$F($**blue** $\wedge X$**green**$)$

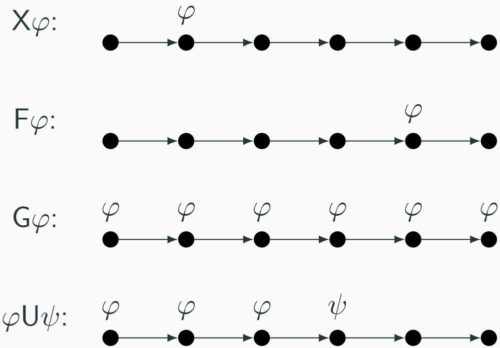


Temporal Logic for traceproperties: Semantics by Example



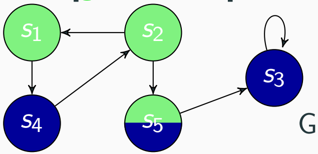


Temporal Logic for traceproperties: Semantics by Example



F(**blue** $\wedge$ X**green**)

[**green**U**blue**]



G**blue**

Examples for Interesting Properties in Temporal Logics

Mutal exclusion, i.e., no two processes can be in their critical section at the same time:

$$\mathsf{G}(\neg p_1 \vee \neg p_2)$$

Starvation freeness, i.e., there is always a call to process p:

$$\mathsf{GF}p$$

Progress, i.e., some property r which implies a future call of process p:

$$\mathsf{G}(r \to \mathsf{F}p)$$

Logics for traceproperties

- Linear-time temporal logic (LTL) is one of the most prominent logics for the specification and verification of reactive and concurrent systems.
- Model checking tools like SPIN and NuSMV automatically verify whether a given computer system is correct with respect to its LTL specification.

Logics for traceproperties

- Linear-time temporal logic (LTL) is one of the most prominent logics for the specification and verification of reactive and concurrent systems.
- Model checking tools like SPIN and NuSMV automatically verify whether a given computer system is correct with respect to its LTL specification.
- One reason for the success of LTL over first-order logic is that LTL is a purely modal logic and thus has many desirable properties.
    - LTL is decidable (PSPACE-comp. model checking and satisfiability) [SC85; CES86].
    - $\mathrm{FO}^2(\leq)$ and $\mathrm{FO}^3(\leq)$ SAT are NEXPTIME-c. and non-elementary [EVW02; Sto74] .
- Caveat: LTL can specify only traceproperties.

Recipe for logics for hyperproperties [Cla+14]

A logic for traceproperties $\rightsquigarrow$ add trace quantifiers

In LTL the satisfying object is a trace: $T \models \varphi$ iff $\forall t \in T : t \models \varphi$

$$\varphi ::= p \mid \neg\varphi \mid (\varphi \vee \varphi) \mid X\varphi \mid \varphi U \varphi$$

In HyperLTL the satisfying object is a set of traces and a trace assignment: $\Pi \models_T \varphi$

$$\varphi ::= \exists\pi\varphi \mid \forall\pi\varphi \mid \psi$$
$$\psi ::= p_\pi \mid \neg\psi \mid (\psi \vee \psi) \mid X\psi \mid \psi U \psi$$

Recipe for logics for hyperproperties [Cla+14]

A logic for traceproperties $\rightsquigarrow$ add trace quantifiers

In LTL the satisfying object is a trace: $T \models \varphi$ iff $\forall t \in T : t \models \varphi$

$$\varphi ::= p \mid \neg\varphi \mid (\varphi \vee \varphi) \mid X\varphi \mid \varphi U \varphi$$

In HyperLTL the satisfying object is a set of traces and a trace assignment: $\Pi \models_T \varphi$

$$\varphi ::= \exists\pi\varphi \mid \forall\pi\varphi \mid \psi$$
$$\psi ::= p_\pi \mid \neg\psi \mid (\psi \vee \psi) \mid X\psi \mid \psi U \psi$$

HyperQPTL extends HyperLTL by (uniform) quantification of propositions: $\exists p\varphi$, $\forall p\varphi$

Hyperlogics via quantifier extensions

- LTL, QPTL, CTL, etc. vs. HyperLTL, HyperQPTL, HyperCTL, etc.
  are prominent logics for traceproperties vs. hyperproperties of systems
  - Traceproperty: Each request is eventually granted (properties of traces)
  - Hyperproperty: Non-inference (values of public outputs do not leak information about confidential bits), (properties of sets of traces)

Hyperlogics via quantifier extensions

- LTL, QPTL, CTL, etc. vs. HyperLTL, HyperQPTL, HyperCTL, etc.
  are prominent logics for traceproperties vs. hyperproperties of systems
  - Traceproperty: Each request is eventually granted (properties of traces)
  - Hyperproperty: Non-inference (values of public outputs do not leak information about confidential bits), (properties of sets of traces)
- HyperLogics are of high complexity or undecidable.
  Not well suited for properties involving unbounded number of traces.

Properties of quantification based hyperproperties

- Quantification based logics for hyperproperties: HyperLTL, HyperCTL, etc.
- Retain some desirable properties of LTL, but are not purely modal logics
  - Model checking for $\exists^*$HyperLTL and HyperLTL are PSPACE and non-elementary [FH16; Cla+14].
  - HyperLTL satisfiability is highly undecidable [For+21].
  - HyperLTL formulae express properties expressible using fixed finite number of traces.

Properties of quantification based hyperproperties

- Quantification based logics for hyperproperties: HyperLTL, HyperCTL, etc.
- Retain some desirable properties of LTL, but are not purely modal logics
  - Model checking for $\exists^*$HyperLTL and HyperLTL are PSPACE and non-elementary [FH16; Cla+14].
  - HyperLTL satisfiability is highly undecidable [For+21].
  - HyperLTL formulae express properties expressible using fixed finite number of traces.
- Bounded termination is not definable in HyperLTL (but is in HyperQPTL)
- Team semantics is a candidate for a purely modal logic without the above caveat.

LTL, HyperLTL, and TeamLTL [MFCS 2018]

In LTL the satisfying object is a trace: $T \models \varphi$ iff $\forall t \in T : t \models \varphi$

$$\varphi ::= p \mid \neg\varphi \mid (\varphi \vee \varphi) \mid X\varphi \mid \varphi U \varphi$$

In HyperLTL the satisfying object is a set of traces and a trace assignment: $\Pi \models_T \varphi$

$$\varphi ::= \exists \pi \varphi \mid \forall \pi \varphi \mid \psi$$
$$\psi ::= p_\pi \mid \neg\psi \mid (\psi \vee \psi) \mid X\psi \mid \psi U \psi$$

In TeamLTL the satisfying object is a set of traces. We use team semantics: $(T, i) \models \varphi$

$$\varphi ::= p \mid \neg p \mid (\varphi \vee \varphi) \mid (\varphi \wedge \varphi) \mid X\varphi \mid \varphi U \mid \varphi W \varphi$$

$+$ new atomic statements (dependence and inclusion atoms: $\mathrm{dep}(\vec{p}, q)$, $\vec{p} \subseteq \vec{q}$)
$+$ additional connectives (Boolean disjunction, contradictory negation, etc.)

Extensions are a well-defined way to delineate expressivity and complexity

Semantics of TeamLTL

Temporal team semantics is universal and synchronous

$$(T, i) \models p \text{ iff } \forall t \in T : p \in t[i] \qquad (T, i) \models \neg p \text{ iff } \forall t \in T : p \notin t[i]$$

Semantics of TeamLTL

Temporal team semantics is universal and synchronous

$$(T, i) \models p \text{ iff } \forall t \in T : p \in t[i] \qquad (T, i) \models \neg p \text{ iff } \forall t \in T : p \notin t[i]$$

$$(T, i) \models \mathsf{F}\varphi \text{ iff } (T, j) \models \varphi \text{ for some } j \geq i \quad (T, i) \models \mathsf{G}\varphi \text{ iff } (T, j) \models \varphi \text{ for all } j \geq i$$

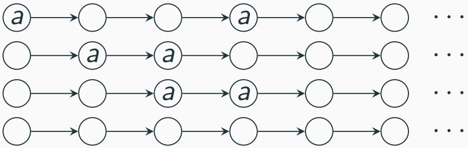
Example: HyperQLTL

There is a timepoint (common for all traces) where a is false in each trace.
Not expressible in HyperLTL, but is in HyperQPTL.

Example: HyperQLTL

There is a timepoint (common for all traces) where a is false in each trace.
Not expressible in HyperLTL, but is in HyperQPTL.

$$\exists p\, \forall \pi\, \mathrm{G}(p \rightarrow \mathrm{X}\mathrm{G}\neg p) \wedge \mathrm{F}(p \wedge \neg a_\pi)$$

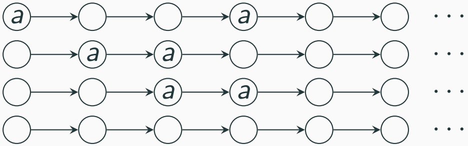


Example: TeamLTL

There is a timepoint (common for all traces) where a is false in each trace.
Not expressible in HyperLTL, but is in HyperQPTL.

$$\exists p \, \forall \pi \, \mathsf{G}(p \rightarrow \mathsf{XG}\neg p) \land \mathsf{F}(p \land \neg a_\pi)$$

Expressible in synchronous TeamLTL: $\mathsf{F}\,\neg a$



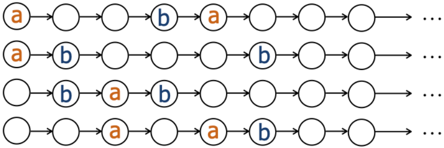
Examples: Disjunction in TeamLTL

A trace-set T satisfies $\varphi \vee \psi$ if it decomposed to sets T_φ and T_ψ satisfying φ and ψ.

$$(T, i) \models \varphi \vee \psi \text{ iff } (T_1, i) \models \varphi \text{ and } (T_2, i) \models \psi, \text{ for some } T_1 \cup T_2 = T$$
$$(T, i) \models \varphi \wedge \psi \text{ iff } (T, i) \models \varphi \text{ and } (T, i) \models \psi$$

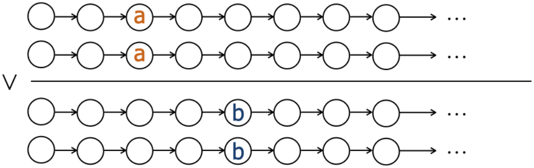
HyperLTL:

$\forall \pi . \forall \pi' .\ \mathrm{F}((a_\pi \wedge a_{\pi'}) \vee (b_\pi \wedge b_{\pi'}))$



TeamLTL:

$(\mathrm{F}\ a) \vee (\mathrm{F}\ b)$



Examples: Dependence atom in TeamLTL

Dependence atom $\mathrm{dep}(p_1, \ldots, p_m, q)$ states that $p_1, \ldots, p_m$ functionally determine q:

$$
\begin{aligned}
(T, i) \models \mathrm{dep}(p_1, \ldots, p_m, q) \;\text{ iff }\; & \forall t, t' \in T \\
& \{p_1, \ldots, p_m\} \cap t[i] = \{p_1, \ldots, p_m\} \cap t'[i] \quad \Rightarrow \quad \{q\} \cap t[i] = \{q\} \cap t'[i]
\end{aligned}
$$

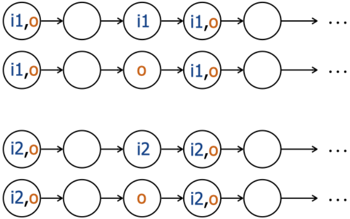
Examples: Dependence atom in TeamLTL

Dependence atom $\mathrm{dep}(p_1, \ldots, p_m, q)$ states that $p_1, \ldots, p_m$ functionally determine q:

$$(T, i) \models \mathrm{dep}(p_1, \ldots, p_m, q) \text{ iff } \forall t, t' \in T$$
$$\{p_1, \ldots, p_m\} \cap t[i] = \{p_1, \ldots, p_m\} \cap t'[i] \quad \Rightarrow \quad \{q\} \cap t[i] = \{q\} \cap t'[i]$$

$(\mathrm{G}\ dep(i1, o)) \vee (\mathrm{G}\ dep(i2, o))$

Nondeterministic dependence: "o either depends on i1 or on i2"



"whenever the traces agree on i1, they agree on o"

$\vee$

"whenever the traces agree on i2, they agree on o"

Inclusion atoms and non-inference

Inclusion atom $(p_1, \ldots, p_n) \subseteq (q_1, \ldots, q_n)$ states that all truth value combinations that occur for $p_1, \ldots, p_n$ also occur for $q_1, \ldots, q_n$:

$$(T, i) \models (p_1, \ldots, p_n) \subseteq (q_1, \ldots, q_n) \text{ iff } \forall t \in T\, \exists s \in T \quad \{p_1, \ldots, p_n\} \cap t[i] = \{p_1, \ldots, p_n\} \cap s[i]$$

Inclusion atoms and non-inference

Inclusion atom $(p_1, \ldots, p_n) \subseteq (q_1, \ldots, q_n)$ states that all truth value combinations that occur for $p_1, \ldots, p_n$ also occur for $q_1, \ldots, q_n$:

$$(T, i) \models (p_1, \ldots, p_n) \subseteq (q_1, \ldots, q_n) \text{ iff } \forall t \in T \, \exists s \in T \quad \{p_1, \ldots, p_n\} \cap t[i] = \{p_1, \ldots, p_n\} \cap s[i]$$

This can be used, e.g, to express non-inference

$$(p_1, \ldots, p_n, s) \subseteq (q_1, \ldots, q_n, \neg s).$$

Public observables $p_1, \ldots, p_n$ do not reveal the secret s.

Temporal team semantics

Definition 1

Temporal team is (T, i), where T a set of traces and $i \in \mathbb{N}$.

$$
\begin{array}{lll}
(T, i) \models p & \text{iff} & \forall t \in T : p \in t[0] \\
(T, i) \models \neg p & \text{iff} & \forall t \in T : p \notin t[0] \\
(T, i) \models \phi \wedge \psi & \text{iff} & (T, i) \models \phi \text{ and } (T, i) \models \psi \\
(T, i) \models \phi \vee \psi & \text{iff} & (T_1, i) \models \phi \text{ and } (T_2, i) \models \psi, \text{ for some } T_1, T_2 \text{ s.t. } T_1 \cup T_2 = T \\
(T, i) \models \mathrm{X}\varphi & \text{iff} & (T, i+1) \models \varphi \\
(T, i) \models \phi \mathrm{U} \psi & \text{iff} & \exists k \geq i \text{ s.t. } (T, k) \models \psi \text{ and } \forall m : i \leq m < k \Rightarrow (T, m) \models \phi \\
(T, i) \models \phi \mathrm{W} \psi & \text{iff} & \forall k \geq i : (T, k) \models \phi \text{ or } \exists m \text{ s.t. } i \leq m \leq k \text{ and } (T, m) \models \psi
\end{array}
$$

Hyperlogic via team semantics

- Temporal logics with team semantics express hyperproperties.
- Purely modal logic & well suited for properties of unbounded number of traces.

Hyperlogic via team semantics

- Temporal logics with team semantics express hyperproperties.
- Purely modal logic & well suited for properties of unbounded number of traces.
- Expressivity
  - TeamLTL and HyperLogics are othogonal in expressivity.
  - Well behaved fragments of TeamLTL can be translated to HyperLogics with some form of set quantification.
  - Upper bound of expressivity is often monadic second-order logic with equi-level predicate.

Hyperlogic via team semantics

- Temporal logics with team semantics express hyperproperties.
- Purely modal logic & well suited for properties of unbounded number of traces.
- Expressivity
  - TeamLTL and HyperLogics are othogonal in expressivity.
  - Well behaved fragments of TeamLTL can be translated to HyperLogics with some form of set quantification.
  - Upper bound of expressivity is often monadic second-order logic with equi-level predicate.
- Complexity landscape is not completely mapped
  - Where is the undecidability frontier of TeamLTL extensions?
  - A large EXPTIME fragment: left-flat and downward closed logics
  - Already TeamLTL with inclusion atoms and Boolean disjunctions is undecidable

Generalised atoms and complete logics

Let B be a set of n-ary Boolean relations. We define the property $[\varphi_1, \ldots, \varphi_n]_B$ for an n-tuple $(\varphi_1, \ldots, \varphi_n)$ of LTL-formulae:

$$(T, i) \models [\varphi_1, \ldots, \varphi_n]_B \quad \text{iff} \quad \{(\llbracket \phi_1 \rrbracket_{(t,i)}, \ldots, \llbracket \phi_n \rrbracket_{(t,i)}) \mid t \in T\} \in B.$$

Generalised atoms and complete logics

Let B be a set of n-ary Boolean relations. We define the property $[\varphi_1, \dots, \varphi_n]_B$ for an n-tuple $(\varphi_1, \dots, \varphi_n)$ of LTL-formulae:

$$(T, i) \models [\varphi_1, \dots, \varphi_n]_B \quad \text{iff} \quad \{(\llbracket \phi_1 \rrbracket_{(t,i)}, \dots, \llbracket \phi_n \rrbracket_{(t,i)}) \mid t \in T\} \in B.$$

Theorem 2 ([FSTTCS 2021])

$\mathrm{TeamLTL}(\varovee, \mathrm{NE}, \overset{1}{\mathrm{A}})$ *can express all* $[\varphi_1, \dots, \varphi_n]_B$.
$\mathrm{TeamLTL}(\varovee, \overset{1}{\mathrm{A}})$ *can express all* $[\varphi_1, \dots, \varphi_n]_B$*, for downward closed* B.

- $(T, i) \models \mathrm{NE}$ iff $T \neq \emptyset$.
- $(T, i) \models \overset{1}{\mathrm{A}}\varphi$ iff $(\{t\}, i) \models \varphi$, for all $t \in T$.

Complexity results

Logic	Model Checking Result
$\mathrm{TeamLTL}$ without $\vee$	in PSPACE [MFCS 2018]
k-coherent $\mathrm{TeamLTL}(\sim)$	in EXPSPACE [FSTTCS 2021]
left-flat $\mathrm{TeamLTL}(\circledvee, \overset{1}{\mathsf{A}})$	in EXPSPACE [FSTTCS 2021]
$\mathrm{TeamLTL}(\subseteq, \circledvee)$	Σ_1^0-hard [FSTTCS 2021]
$\mathrm{TeamLTL}(\subseteq, \circledvee, \mathsf{A})$	Σ_1^1-hard [FSTTCS 2021]
$\mathrm{TeamLTL}(\sim)$	complete for third-order arithmetic [Lüc20]

- k-coherence: $(T, i) \models \varphi$ iff $(S, i) \models \varphi$ for all $S \subseteq T$ s.t. $|S| \leq k$
- left-flatness: Restrict U and W syntactically to $(\overset{1}{\mathsf{A}}\varphi \mathsf{U} \psi)$ and $(\overset{1}{\mathsf{A}}\varphi \mathsf{W} \psi)$
- $\sim$ is contradictory negation and $\mathrm{TeamLTL}(\sim)$ subsumes all the other logics

Source of Undecidability

Definition 3

A non-deterministic 3-counter machine M consists of a list I of n instructions that manipulate three counters C_l, C_m and C_r. All instructions are of the following forms:

- C_a^+ goto $\{j_1, j_2\}$, $\quad C_a^-$ goto $\{j_1, j_2\}$, $\quad$ if $C_a = 0$ goto $j_1$else goto j_2,

where $a \in \{l, m, r\}$, $0 \leq j_1, j_2 < n$.

Source of Undecidability

Definition 3

A non-deterministic 3-counter machine M consists of a list I of n instructions that manipulate three counters C_l, C_m and C_r. All instructions are of the following forms:

- C_a^+ goto $\{j_1, j_2\}$, $\qquad C_a^-$ goto $\{j_1, j_2\}$, $\qquad$ if $C_a = 0$ goto $j_1$else goto j_2,

where $a \in \{l, m, r\}$, $0 \leq j_1, j_2 < n$.

- configuration: tuple (i, j, k, l), where $0 \leq i < n$ is the next instruction to be executed, and $j, k, l \in \mathbb{N}$ are the current values of the counters C_l, C_m and C_r.
- computation: infinite sequence of consecutive configurations starting from the initial configuration $(0, 0, 0, 0)$.
- computation b-recurring if the instruction labelled b occurs infinitely often in it.
- computation is lossy if the counter values can non-deterministically decrease

Undecidability results

Theorem 4 ([AH94; Sch10])

Deciding whether a given non-deterministic 3-counter machine has a (lossy) b-recurring computation for a given b is (Σ_1^0-complete) Σ_1^1-complete.

Undecidability results

Theorem 4 ([AH94; Sch10])

Deciding whether a given non-deterministic 3-counter machine has a (lossy) b-recurring computation for a given b is (Σ^0_1-complete) Σ^1_1-complete.

Theorem 5 ([FSTTCS 2021])

Model checking for $\mathrm{TeamLTL}(\varovee, \subseteq)$ *is* Σ^1_0*-hard.*
Model checking for $\mathrm{TeamLTL}(\varovee, \subseteq, \mathsf{A})$ *is* Σ^1_1*-hard.*

Proof Idea:

- reduce existence of b-recurring computation of given 3-counter machine M and instruction label b to model checking problem of $\mathrm{TeamLTL}(\varovee, \subseteq, \mathsf{A})$
- $\mathrm{TeamLTL}(\varovee, \subseteq)$ suffices to enforce lossy computation
- $T[i, \infty]$ encodes the value of counters of the ith configuration
  the value of C_a is the cardinality of the set $\{t \in T[i, \infty] \mid c_a \in t[0]\}$

Model checking for $\mathrm{TeamLTL}(\subseteq, \varovee)$ is Σ^0_1-hard.

Proof.

Given a set I of instructions of a 3-counter machine M, and an instruction label b, we construct a $\mathrm{TeamLTL}(\subseteq, \varovee)$-formula $\varphi_{I,b}$ and a Kripke structure $\mathfrak{K}_I$ such that

$$(\mathrm{Traces}(\mathfrak{K}_I), 0) \models \varphi_{I,b} \quad \text{iff} \quad M \text{ has a } b\text{-recurring lossy computation.} \tag{1}$$

The Σ^0_1-hardness then follows since the construction is computable. □

Idea of the encoding

A set of traces T encodes the configuration $(c, d, e, f) \in \mathbb{N}^4$, if

- For all $t \in T$, the only instruction in $t[0]$ is c.
- The cardinality of $\{t \in T \mid c_l \in t[0]\}$ is d.
- The cardinality of $\{t \in T \mid c_m \in t[0]\}$ is e.
- The cardinality of $\{t \in T \mid c_r \in t[0]\}$ is f.

Idea of the encoding

A set of traces T encodes the configuration $(c, d, e, f) \in \mathbb{N}^4$, if

- For all $t \in T$, the only instruction in $t[0]$ is c.
- The cardinality of $\{t \in T \mid c_l \in t[0]\}$ is d.
- The cardinality of $\{t \in T \mid c_m \in t[0]\}$ is e.
- The cardinality of $\{t \in T \mid c_r \in t[0]\}$ is f.

The infinite sequence $(T[i, \infty])_{i \in \mathbb{N}}$ encodes an infinite computation.

Idea of the encoding

A set of traces T encodes the configuration $(c, d, e, f) \in \mathbb{N}^4$, if

- For all $t \in T$, the only instruction in $t[0]$ is c.
- The cardinality of $\{t \in T \mid c_l \in t[0]\}$ is d.
- The cardinality of $\{t \in T \mid c_m \in t[0]\}$ is e.
- The cardinality of $\{t \in T \mid c_r \in t[0]\}$ is f.

The infinite sequence $(T[i, \infty])_{i \in \mathbb{N}}$ encodes an infinite computation.

The computation is lossy, since distinct traces in T may collapse to a single trace in $T[i, \infty]$.

Idea of the encoding

A set of traces T encodes the configuration $(c, d, e, f) \in \mathbb{N}^4$, if

- For all $t \in T$, the only instruction in $t[0]$ is c.
- The cardinality of $\{t \in T \mid c_l \in t[0]\}$ is d.
- The cardinality of $\{t \in T \mid c_m \in t[0]\}$ is e.
- The cardinality of $\{t \in T \mid c_r \in t[0]\}$ is f.

The infinite sequence $(T[i, \infty])_{i \in \mathbb{N}}$ encodes an infinite computation.

The computation is lossy, since distinct traces in T may collapse to a single trace in $T[i, \infty]$.

The Kripke structure $\mathrm{Traces}(\mathfrak{K}_I)$ encodes all infinite sequences of configurations.

Construction of the formula

The formula $\phi_{I,b}$ enforces that the configurations encoded by $T[i,\infty]$, $i \in \mathbb{N}$, encode an accepting computation of the counter machine; $\vee_{\mathrm{L}}$ guesses the computation.

$$\phi_{I,b} \coloneqq (\theta_{\mathrm{comp}} \wedge \theta_{b-\mathrm{rec}}) \vee_{\mathrm{L}} \top.$$

The formula θ_{comp} states that the encoded computation is legal.

Construction of the formula

The formula $\phi_{I,b}$ enforces that the configurations encoded by $T[i,\infty]$, $i \in \mathbb{N}$, encode an accepting computation of the counter machine; $\vee_{\mathrm{L}}$ guesses the computation.

$$\phi_{I,b} \coloneqq (\theta_{\mathrm{comp}} \wedge \theta_{b-\mathrm{rec}}) \vee_{\mathrm{L}} \top.$$

The formula θ_{comp} states that the encoded computation is legal.

The formula

$$\theta_{b-\mathrm{rec}} \coloneqq \mathsf{GF}b$$

describes the b-recurrence condition of the computation.

Expressing legality of computation

Define

$$\text{singleton} := \mathsf{G} \bigwedge_{a \in \mathrm{PROP}} (a \varovee \neg a), \qquad \mathsf{c_s}\text{-non-increase} := c_s \vee (\neg c_s \wedge \mathsf{X} \neg c_s), \text{ for } s \in \{l, m, r\}.$$

Expressing legality of computation

Define

$$\text{singleton} := \mathsf{G} \bigwedge_{a \in \mathrm{PROP}} (a \varovee \neg a), \qquad \mathsf{c_s}\text{-non-increase} := c_s \vee (\neg c_s \wedge \mathsf{X} \neg c_s), \text{ for } s \in \{l, m, r\}.$$

For the instruction $i\colon\ C_l^+$ goto $\{j, j'\}$, define

$$\theta_i := \mathsf{X}(j \varovee j') \wedge \big((\text{singleton} \wedge \neg c_l \wedge \mathsf{X} c_l) \vee \mathsf{c_l}\text{-non-increase}\big) \wedge \mathsf{c_r}\text{-non-increase} \wedge \mathsf{c_m}\text{-non-increase}\,.$$

Expressing legality of computation

Define

$$\text{singleton} := \mathsf{G} \bigwedge_{a \in \mathrm{PROP}} (a \oslash \neg a), \qquad \mathsf{c_s}\text{-non-increase} := c_s \vee (\neg c_s \wedge \mathsf{X}\neg c_s), \text{ for } s \in \{l, m, r\}.$$

For the instruction i: C_l^+ goto $\{j, j'\}$, define

$$\theta_i := \mathsf{X}(j \oslash j') \wedge \big((\text{singleton} \wedge \neg c_l \wedge \mathsf{X} c_l) \vee \mathsf{c_l}\text{-non-increase}\big) \wedge \mathsf{c_r}\text{-non-increase} \wedge \mathsf{c_m}\text{-non-increase}\,.$$

For the instruction i: if $C_s = 0$ goto j, else goto j', defined similarly.

Expressing legality of computation

Define

$$\text{singleton} := \mathsf{G} \bigwedge_{a \in \mathrm{PROP}} (a \oslash \neg a), \qquad \mathsf{c_s}\text{-non-increase} := c_s \vee (\neg c_s \wedge \mathsf{X} \neg c_s),\ \text{for } s \in \{l, m, r\}.$$

For the instruction $i\colon\ C_l^+$ goto $\{j, j'\}$, define

$$\theta_i := \mathsf{X}(j \oslash j') \wedge \big((\text{singleton} \wedge \neg c_l \wedge \mathsf{X} c_l) \vee \mathsf{c_l}\text{-non-increase}\,\big) \wedge \mathsf{c_r}\text{-non-increase} \wedge \mathsf{c_m}\text{-non-increase}\,.$$

For the instruction i: if $C_s = 0$ goto j, else goto j', defined similarly.

Finally, define $\theta_{\mathrm{comp}} := \mathsf{G} \bigoslash_{i<n} (i \wedge \theta_i)$.

Known complexity results

Logic	TSAT	TPC	TMC
LTL	PSPACE	PSPACE	PSPACE-hard
$\mathrm{LTL}(\mathrm{dep})$	PSPACE	PSPACE	NEXPTIME-hard
$\mathrm{LTL}(\otimes, \mathcal{D})$	Σ^0_1-hard	PSPACE	Σ^0_1-hard
$\mathrm{TeamLTL}(\subseteq, \otimes)$	Σ^0_1-hard	?	Σ^0_1-hard
$\mathrm{TeamLTL}(\subseteq, \otimes, \overset{1}{\mathsf{A}})$	Σ^1_1-hard	?	Σ^1_1-hard
$\mathrm{LTL}(\mathcal{D}, \sim)$	third-order arithmetic [Lüc20]	PSPACE	third-order arithmetic [Lüc20]
$\mathrm{LTL} - \vee$	?	?	$\in$ PSPACE
k-coherent $\mathrm{TeamLTL}(\sim)$	?	?	in EXPSPACE
left-flat $\mathrm{TeamLTL}(\otimes, \overset{1}{\mathsf{A}})$	?	?	in EXPSPACE

Figure 1: Overview of complexity results for TeamLTL. 'dep' refers to dependence atoms, '$\sim$' refers to the contradictory negation, $\mathcal{D}$ refers to any finite set of first-order definable generalised atoms, and '$\mathrm{LTL} - \vee$' refers to disjunction free LTL. References: [MFCS 2018; FSTTCS 2021].

Further related work

- All logics mentioned specify syncronous hyperproperties.
  Attention shifted to logics utilising forms of asynchronicity [LICS 2022; MFCS 2023].
- Logics for quantitative or probabilistic hyperproperties.
  E.g., hyperproperties of Markov decision processes.

Further related work

- All logics mentioned specify syncronous hyperproperties.
  Attention shifted to logics utilising forms of asynchronicity [LICS 2022; MFCS 2023].
- Logics for quantitative or probabilistic hyperproperties.
  E.g., hyperproperties of Markov decision processes.
- Theoretical results are mainly complexity theoretic and expressivity comparisons with variants of MSO.
- Tool support: Automata-based model checker AutoHyper [BF23].

Conclusion

- Introduction into Temporal Logics
- Hyperproperties and Temporal Team Semantics
- Undecidability of model checking of $\mathrm{TeamLTL}(\varovee, \subseteq)$

Bibliography i

[FSTTCS 2021] Jonni Virtema, Jana Hofmann, Bernd Finkbeiner, Juha Kontinen and Fan Yang. 'Linear-Time Temporal Logic with Team Semantics: Expressivity and Complexity'. In: *FSTTCS*. Vol. 213. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, 52:1–52:17.

[LICS 2022] Jens Oliver Gutsfeld, Arne Meier, Christoph Ohrem and Jonni Virtema. 'Temporal Team Semantics Revisited'. In: *LICS '22: 37th Annual ACM/IEEE Symposium on Logic in Computer Science, Haifa, Israel, August 2 - 5, 2022*. Ed. by Christel Baier and Dana Fisman. ACM, 2022, 44:1–44:13. DOI: 10.1145/3531130.3533360.

[MFCS 2018] Andreas Krebs, Arne Meier, Jonni Virtema and Martin Zimmermann. 'Team Semantics for the Specification and Verification of Hyperproperties'. In: *MFCS*. Vol. 117. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018, 10:1–10:16.

[MFCS 2023] Juha Kontinen, Max Sandström and Jonni Virtema. 'Set Semantics for Asynchronous TeamLTL: Expressivity and Complexity'. In: *MFCS*. Vol. 272. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023, 60:1–60:14.

[AH94] Rajeev Alur and Thomas A. Henzinger. 'A Really Temporal Logic'. In: *J. ACM* 41.1 (1994), pp. 181–204.

[BF23] Raven Beutner and Bernd Finkbeiner. 'AutoHyper: Explicit-State Model Checking for HyperLTL'. In: *TACAS (1)*. Vol. 13993. Lecture Notes in Computer Science. Springer, 2023, pp. 145–163.

[CES86] E. Clarke, E. Allen Emerson and A. Sistla. 'Automatic Verification of Finite-State Concurrent Systems Using Temporal Logic Specifications'. In: *ACM Transactions on Programming Languages and Systems* 8.2 (1986), pp. 244–263.

[Cla+14] Michael R. Clarkson, Bernd Finkbeiner, Masoud Koleini, Kristopher K. Micinski, Markus N. Rabe and César Sánchez. 'Temporal Logics for Hyperproperties'. In: *POST*. Vol. 8414. Lecture Notes in Computer Science. Springer, 2014, pp. 265–284.

[EVW02] Kousha Etessami, Moshe Y. Vardi and Thomas Wilke. 'First-Order Logic with Two Variables and Unary Temporal Logic'. In: *Inf. Comput.* 179.2 (2002), pp. 279–295.

[FH16] Bernd Finkbeiner and Christopher Hahn. 'Deciding Hyperproperties'. In: *CONCUR*. Vol. 59. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016, 13:1–13:14.

Bibliography iv

[For+21] Marie Fortin, Louwe B. Kuijer, Patrick Totzke and Martin Zimmermann. 'HyperLTL Satisfiability Is Σ_1^1-Complete, HyperCTL* Satisfiability Is Σ_1^2-Complete'. In: *MFCS*. Vol. 202. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, 47:1–47:19.

[Lüc20] Martin Lück. 'On the complexity of linear temporal logic with team semantics'. In: *Theor. Comput. Sci.* 837 (2020), pp. 1–25.

[SC85] A. Prasad Sistla and Edmund M. Clarke. 'The Complexity of Propositional Linear Temporal Logics'. In: *J. ACM* 32.3 (1985), pp. 733–749.

[Sch10] Philippe Schnoebelen. 'Lossy Counter Machines Decidability Cheat Sheet'. In: *RP*. Vol. 6227. Lecture Notes in Computer Science. Springer, 2010, pp. 51–75.

Bibliography v

[Sto74] L.J. Stockmeyer. *The Complexity of Decision Problems in Automata Theory and Logic*. MAC TR. Massachusetts Institute of Technology, Project MAC, 1974. URL: https://books.google.co.uk/books?id=zFbQMQAACAAJ.

Extra complexity result

Proposition 6

How complicated it is to decide whether a $\mathrm{TeamLTL}(\subseteq, \varovee)$*-formula is 1-coherent?*

Extra complexity result

Proposition 6

Deciding whether a $\mathrm{TeamLTL}(\subseteq, \varovee)$*-formula is 1-coherent is* Π^0_1*-hard.*

Extra complexity result

Proposition 6

Deciding whether a $\mathrm{TeamLTL}(\subseteq, \varovee)$*-formula is 1-coherent is* Π^0_1*-hard.*

Proof.

Input: $\mathrm{TeamLTL}(\subseteq, \varovee)$-formula φ.

1. Rewrite φ into an LTL-formula φ^* equivalent to φ over singleton teams.
2. φ is not satisfiable, if and only if, φ 1-coherent and φ^* is not satisfiable.
3. Since deciding whether φ^* is not satisfiable is done in PSPACE and deciding whether φ is not satisfiable is Π^0_1-hard, checking 1-coherence of φ is Π^0_1-hard.

$\square$

Extra complexity result

Proposition 6

Deciding whether a $\mathrm{TeamLTL}(\subseteq, \varovee)$-formula is 1-coherent is Π^0_1-hard.

Proof.

Input: $\mathrm{TeamLTL}(\subseteq, \varovee)$-formula φ.

1. Rewrite φ into an LTL-formula φ^* equivalent to φ over singleton teams.
2. φ is not satisfiable, if and only if, φ 1-coherent and φ^* is not satisfiable.
3. Since deciding whether φ^* is not satisfiable is done in PSPACE and deciding whether φ is not satisfiable is Π^0_1-hard, checking 1-coherence of φ is Π^0_1-hard.

If φ is not satisfiable, then trivially it is 1-coherent and φ^* is not satisfiable.
If φ is 1-coherent then it is safisfiable, if and only if, it is satisfiable on a singleton team. Hence if φ^* is not satisfiable then neither is φ. $\square$