Complexity and Expressivity of Propositional Logics with Team Semantics
Arne Meier, Jonni Virtema
9th of August

Lecture 5: Recent Trends: Hyperproperties

Literature: [Vir+21; Gut+22]

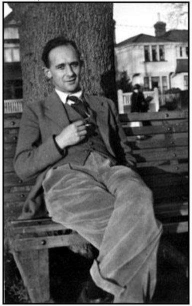
Temporal Logic

Modal logic : $\Box\varphi$, $\Diamond\varphi$

$\forall w'$ s.t. $\exists w'$

wRw'

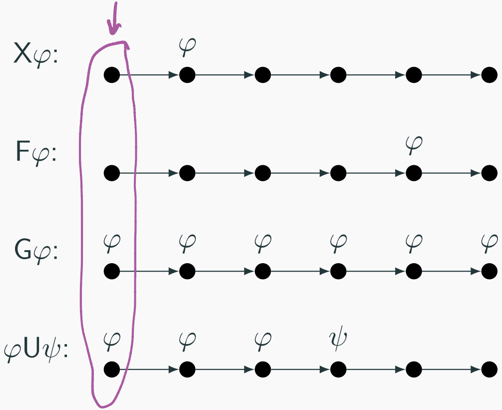
- Dates back to Arthur Norman Prior (1914–1969)
- New modalities neXt, Until, Future, Global
- and quantifiers: All paths, Exists a path
- 'From Philosophical to Industrial Logics' [Var09]

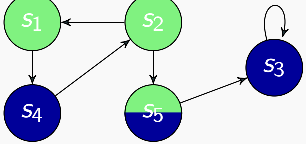


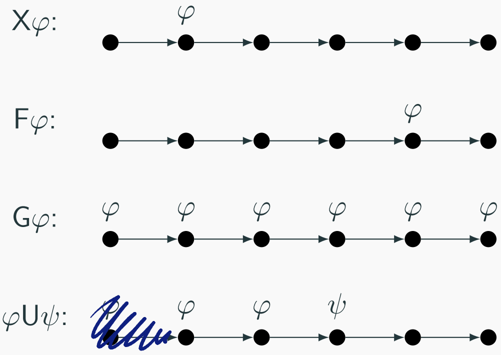
Arthur N. Prior
(1914–1969)

Temporal Logic: Semantics by Example

the present world





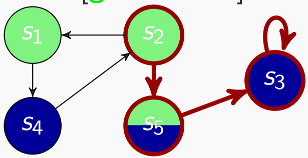


s_2 s_5 s_3 ...

green green blue

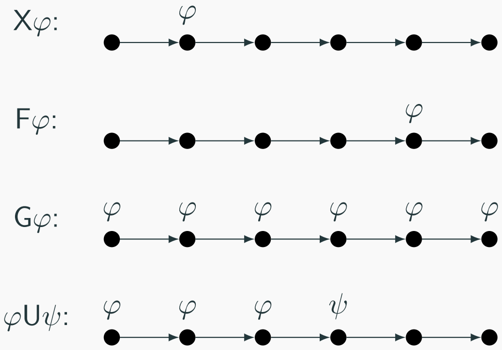
there exists a path s.t. green U blue is true.

E[greenUblue]

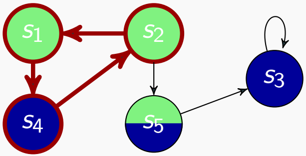


$s_2, s_5, s_3, s_3, \ldots$

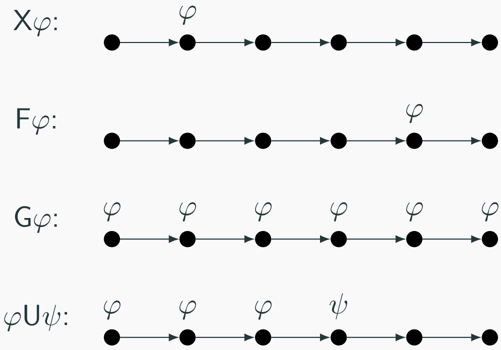
Temporal Logic: Semantics by Example

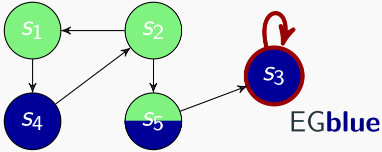


$\mathrm{EF}(\textbf{blue} \wedge \mathrm{AX}\textbf{green})$

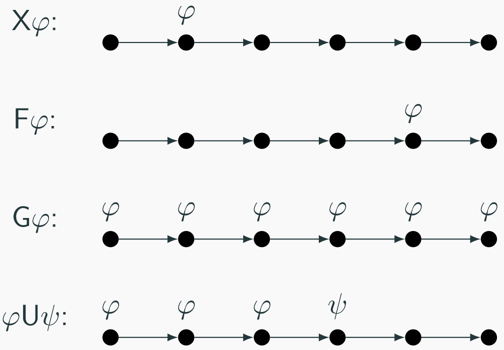


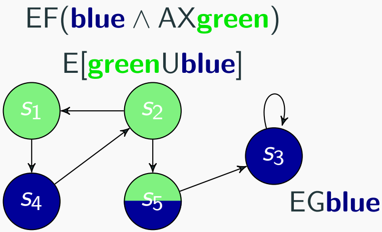
Temporal Logic: Semantics by Example





Temporal Logic: Semantics by Example

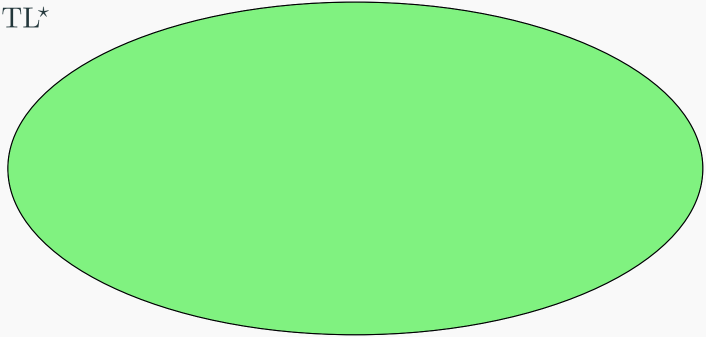




A Temporal Lanscape of Logics

Full branching time logic

$\mathrm{CTL}^\star$

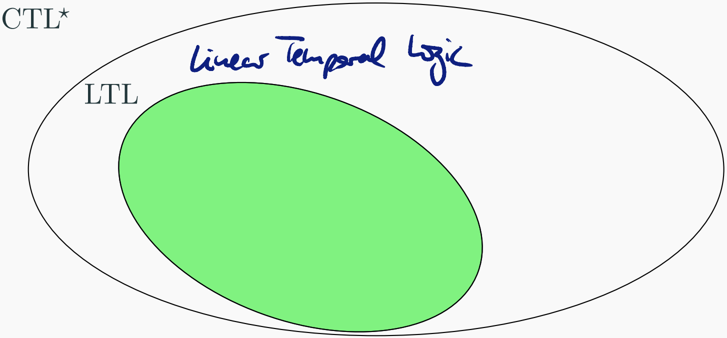


$\mathrm{CTL}^\star$: Syntax

$$\varphi ::= \top \mid x \mid \varphi \wedge \varphi \mid \neg\varphi \mid \mathsf{X}\varphi \mid \varphi\mathsf{U}\varphi \mid \mathsf{E}\varphi,$$

where $x \in \mathrm{PROP}$.

A Temporal Lanscape of Logics



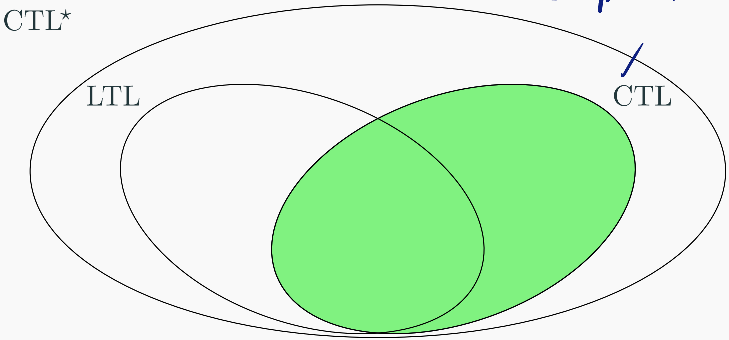
LTL: Formulae $E\varphi$ with

$$\varphi ::= \top \mid x \mid \varphi \wedge \varphi \mid \neg\varphi \mid X\varphi \mid \varphi U \varphi,$$

where $x \in \mathrm{PROP}$.

A Temporal Lanscape of Logics

Computation Tree Logic



CTL: Syntax

$$\varphi ::= \top \mid x \mid \varphi \wedge \varphi \mid \neg\varphi \mid \mathsf{EX}\varphi \mid \mathsf{E}[\varphi \mathsf{U} \varphi] \mid \mathsf{AF}\varphi,$$

where $x \in \mathsf{PROP}$.

CTL+

ECTL / ECTL+

State and Path Formulae

Regarding the formulae of $\mathrm{CTL}^\star$, we follow the terminology of Emerson and Sistla [ES84].

S1: Any atomic proposition is a state formula.

S2: If ψ, φ are state formula, then so are $\varphi \wedge \psi$, and $\neg\psi$.

S3: If ψ is a path formula, then Eψ is a state formula.

State and Path Formulae

Regarding the formulae of $\mathrm{CTL}^{\star}$, we follow the terminology of Emerson and Sistla [ES84].

S1: Any atomic proposition is a state formula.

S2: If ψ, φ are state formula, then so are $\varphi \wedge \psi$, and $\neg\psi$.

S3: If ψ is a path formula, then $\mathsf{E}\psi$ is a state formula.

P1: Any state formula is a path formula.

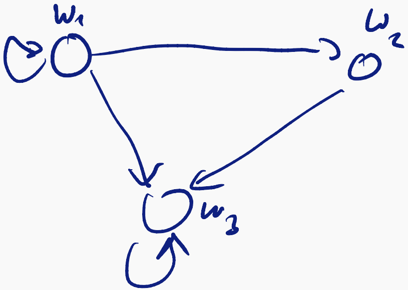
P2: If ψ, φ are path formulae, then so are $\varphi \wedge \psi$, $\neg\psi$.

P3: If ψ, φ are path formulae, then so are $\mathsf{X}\varphi$, $[\varphi \mathsf{U} \psi]$.

Intuitively, (S1), (P1), (P2), and (P3) form LTL.

Definition 48

A frame $\mathcal{F}$ is a tuple $\mathcal{F} = (W, R)$, where W is a set of worlds and R is its transition relation, i.e., $R \subseteq W \times W$ and R is total (every state has ≤ 1 successor).



$$W = \{w_1, w_2, w_3\}$$

$$R = \{(w_1, w_2), (w_2, w_3), (w_1, w_1), (w_3, w_3)\}$$

Kripke Frames and Paths

Definition 48

A frame $\mathcal{F}$ is a tuple $\mathcal{F} = (W, R)$, where W is a set of worlds and R is its transition relation, i.e., $R \subseteq W \times W$ and R is total (every state has ≤ 1 successor).

Definition 49

Let PROP be an countably infinite set of symbols. A model is a tuple $\mathcal{M} = (\mathcal{F}, V)$, where $\mathcal{F} = (W, R)$ is a frame and $V \colon \mathrm{PROP} \to \mathfrak{P}(W)$ is a labeling function.

Definition 50

A path $\pi = w_0, w_1, \ldots$ in a frame (W, R) is an infinite sequence of states such that $(w_i, w_{i+1}) \in R$ for all i. For $\pi = \quad w_1, \ldots$, let $\pi_i := w_i w_{i+1} \ldots$, $\pi[i] := w_i$.

Definition 51

Let $\mathcal{M} = (W, R, V)$ be a model, $w \in W$, π be a path in (W, R).

$$\left.\begin{aligned}
\mathcal{M}, w &\models p \text{ iff } w \in V(p),\\
\mathcal{M}, w &\models \neg\varphi \text{ iff } \mathcal{M}, w \not\models \varphi,\\
\mathcal{M}, w &\models \varphi \wedge \psi \text{ iff } \mathcal{M}, w \models \varphi \text{ and } \mathcal{M}, w \models \psi,\\
\mathcal{M}, w &\models \mathsf{E}\varphi \text{ iff there is a path } \pi \text{ with } \pi[0] = w \text{ we have that } \mathcal{M}, \pi \models \varphi,
\end{aligned}\right\} \text{for state formulae}$$

Definition 51

Let $\mathcal{M} = (W, R, V)$ be a model, $w \in W$, π be a path in (W, R).

$$
\begin{aligned}
\mathcal{M}, w \models p &\text{ iff } w \in V(p),\\
\mathcal{M}, w \models \neg\varphi &\text{ iff } \mathcal{M}, w \not\models \varphi,\\
\mathcal{M}, w \models \varphi \wedge \psi &\text{ iff } \mathcal{M}, w \models \varphi \text{ and } \mathcal{M}, w \models \psi,\\
\mathcal{M}, w \models \mathsf{E}\varphi &\text{ iff there is a path } \pi \text{ with } \pi[0] = w \text{ we have that } \mathcal{M}, \pi \models \varphi,\\
\mathcal{M}, \pi \models p &\text{ iff } \pi[0] \in V(p),\\
\mathcal{M}, \pi \models \neg\varphi &\text{ iff } \mathcal{M}, \pi \not\models \varphi,\\
\mathcal{M}, \pi \models \varphi \wedge \psi &\text{ iff } \mathcal{M}, \pi \models \varphi \text{ and } \mathcal{M}, \pi \models \psi,\\
\mathcal{M}, \pi \models \mathsf{X}\varphi &\text{ iff } \mathcal{M}, \pi_1 \models \varphi,\\
\mathcal{M}, \pi \models \varphi \mathsf{U} \psi &\text{ iff there is a } k \geq 0 \text{ s.t. } \mathcal{M}, \pi[k] \models \psi \text{ and}\\
&\quad \text{for all } i \leq k \text{ we have that } \mathcal{M}, \pi[i] \models \varphi.
\end{aligned}
$$

for path formulae

Remaning Operators by Short Cuts

There exist further operators which can be defined by the operators we have already defined:

$$\begin{aligned} \mathsf{A}\varphi &:= \neg \mathsf{E} \neg \varphi \\ \mathsf{F}\varphi &:= [\top \mathsf{U} \varphi] \\ \mathsf{G}\varphi &:= \neg \mathsf{F} \neg \varphi \\ [\varphi \mathsf{W} \psi] &:= \neg [\neg \varphi \mathsf{U} \neg \psi] \end{aligned}$$

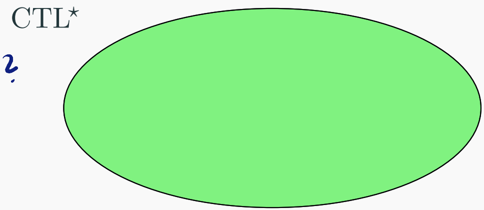
Satisfiability ($\mathrm{CTL}^\star$-SAT)

Given: $\mathrm{CTL}^\star$-formula φ.

Question: Is φ satisfiable? Is there a model $\mathfrak{M}$ s.t. $\mathfrak{M}, w \models \varphi$? $w \in \mathfrak{M}$

Complexity: EEXP-complete [KV85; EJ99]

$\mathrm{TIME}(2^{2^{n^{O(1)}}})$



	SAT	MC
$\mathrm{CTL}^\star$	EEXP	
LTL		
CTL		

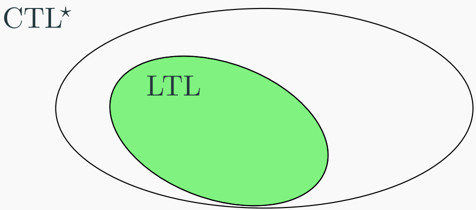
Two Important Problems and their Complexities

Satisfiability (LTL-SAT)

Given: LTL-formula φ.

Question: Is φ satisfiable?

Complexity: PSPACE-complete [SC85]



	SAT	MC
CTL*	EEXP	
LTL	PSPACE	
CTL		

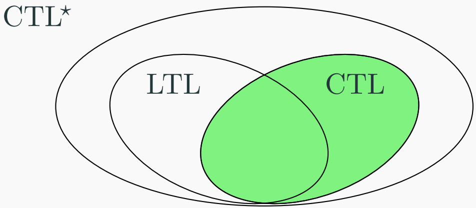
Two Important Problems and their Complexities

Satisfiability (CTL-SAT)

Given: CTL-formula φ.

Question: Is φ satisfiable?

Complexity: EXP-complete [FL79; Pra80]



	SAT	MC
CTL*	EEXP	
LTL	PSPACE	
CTL	EXP	

Two Important Problems and their Complexities

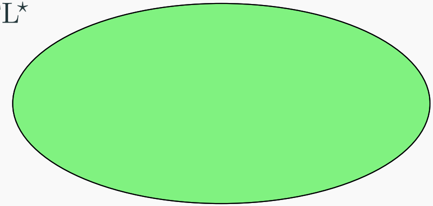
Model Checking ($\mathrm{CTL}^\star$-MC)

Given: $\mathrm{CTL}^\star$-formula φ, model $\mathcal{M}$.

Question: Is there a $w \in \mathcal{M}$ that satisfies φ?

Complexity: PSPACE-complete [CES86]

$\mathrm{CTL}^\star$



	SAT	MC
$\mathrm{CTL}^\star$	EEXP	PSPACE
LTL	PSPACE	
CTL	EXP	

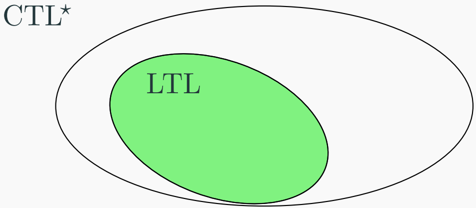
Two Important Problems and their Complexities

Model Checking (LTL-MC)

Given: LTL-formula φ, model $\mathcal{M}$.

Question: Is there a $w \in \mathcal{M}$ that satisfies φ?

Complexity: PSPACE-complete [CES86]



	SAT	MC
CTL*	EEXP	PSPACE
LTL	PSPACE	PSPACE
CTL	EXP	

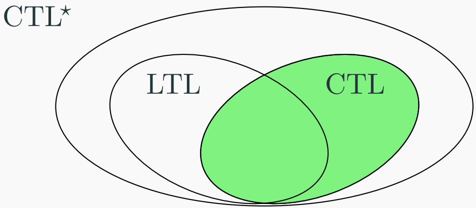
Two Important Problems and their Complexities

Model Checking (CTL-MC)

Given: CTL-formula φ, model $\mathcal{M}$.

Question: Is there a $w \in \mathcal{M}$ that satisfies φ?

Complexity: P-complete [CES86; Sch02]



	SAT	MC
CTL*	EEXP	PSPACE
LTL	PSPACE	PSPACE
CTL	EXP	P

Examples for Interesting Properties in Temporal Logics

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

Global property ⇝ $\underline{AG}\varphi$

Examples for Interesting Properties in Temporal Logics

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

$$\mathrm{AG}(\neg p_1 \vee \neg p_2)$$

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

Examples for Interesting Properties in Temporal Logics

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

$$\mathrm{AG}(\neg p_1 \vee \neg p_2)$$

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

$$\mathrm{AF}p$$

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

Examples for Interesting Properties in Temporal Logics

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

$$AG(\neg p_1 \vee \neg p_2)$$

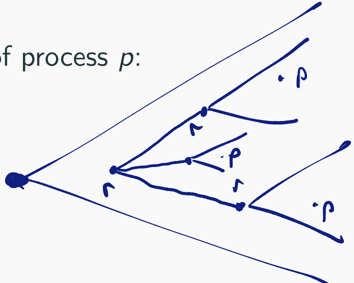
Starvation freeness, i.e., there is always a call to process p:

$$AFp$$

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

$$AG(r \rightarrow AFp)$$

usual shortcut



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.

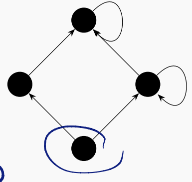
Traceproperties and hyperproperties

Opening your office computer after holidays:

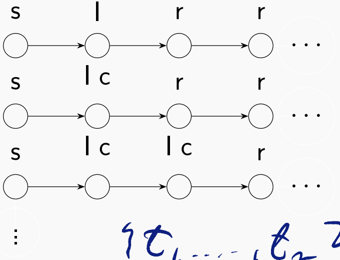
LTL

$A\ F$ READY

t_n is ready in n steps



n



F READY

$\{t_1, \ldots, t_n\}$

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.

$\{t_i \mid i \in \mathbb{N}\}$

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-complete model checking and satisfiability).
    - $\mathrm{FO}^2(\leq)$ and $\mathrm{FO}^3(\leq)$ SAT are NEXPTIME-complete and non-elementary.
- Caveat: LTL can specify only traceproperties.

Recipe for logics for hyperproperties

Π is a trace assignment

$\Pi : VAR \to T$

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: $\Pi \models_T \varphi$

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

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

Hyperlogics via quantifier extensions

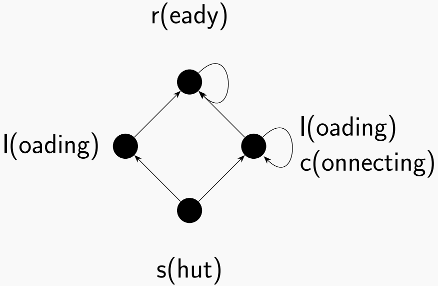
- LTL, QPTL, CTL, etc. vs. HyperLTL, HyperQPTL, HyperCTL, etc. are prominent logics for traceproperties vs. hyperproperties of systems
  - Traceproperty: Each request is eventually granted (properties of traces)
  - Hyperproperty: Non-inference (values of public outputs do not leak information about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

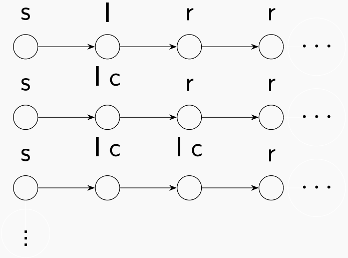
Properties of quantification based hyperproperties

- 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.
  - HyperLTL satisfiability is highly undecidable.
  - 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.
    - HyperLTL satisfiability is highly undecidable. Σ_1^1 - hard
    - HyperLTL formulae express properties expressible using fixed finite number of traces.
- Bounded termination is not definable in HyperLTL (but is in HyperQPTL)





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:
  - 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

LTL, HyperLTL, and TeamLTL

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

$$(T, i) \models p \text{ iff } \forall t \in T : t[i](p) = 1 \qquad (T, i) \models \neg p \text{ iff } \forall t \in T : t[i](p) = 0$$

Semantics of TeamLTL

Temporal team semantics is universal and synchronous

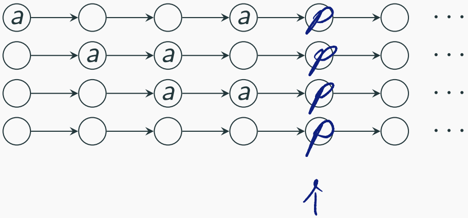
$$(T, i) \models p \text{ iff } \forall t \in T : t[i](p) = 1 \qquad (T, i) \models \neg p \text{ iff } \forall t \in T : t[i](p) = 0$$

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

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.

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

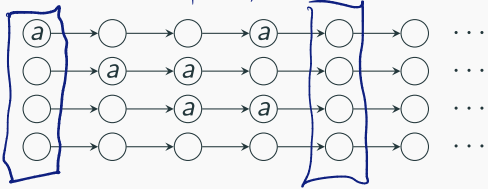


Example: TeamLTL

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

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

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



$T \models F \neg a$ $\qquad$ $T' \models \neg$

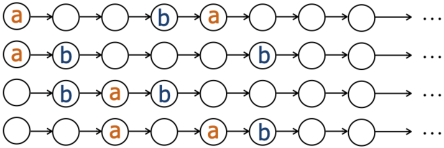
Examples: Disjunction in TeamLTL

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

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

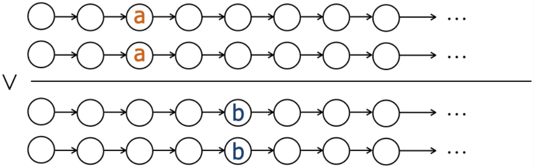
HyperLTL:

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



TeamLTL:

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



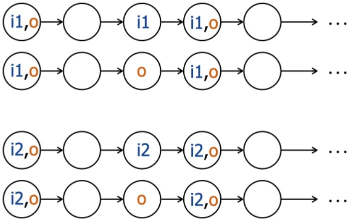
Examples: Dependence atom in TeamLTL

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

$$(T, i) \models \mathrm{dep}(p_1, \ldots, p_m, q) \text{ iff } \forall t, t' \in T \Big(\bigwedge_{1 \leq j \leq m} t[i](p_j) = t'[i](p_j) \Big) \Rightarrow (t[i](q) = t'[i](q))$$

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

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"

Temporal team semantics

Definition 52

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

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

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 $[\varphi_1, \ldots, \varphi_n]_B$ for an n-tuple $(\varphi_1, \ldots, \varphi_n)$ of LTL-formulae:

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

Theorem 53

$\mathrm{TeamLTL}(\varovee, \mathrm{NE}, \overset{1}{\mathsf{A}})$ *can express all* $[\varphi_1, \ldots, \varphi_n]_B$.

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

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

Complexity results

Logic	Model Checking Result
$\mathrm{TeamLTL}$ without $\vee$	in PSPACE
k-coherent $\mathrm{TeamLTL}(\sim)$	in EXPSPACE
left-flat $\mathrm{TeamLTL}(\owedge, \overset{1}{\mathsf{A}})$	in EXPSPACE
$\mathrm{TeamLTL}(\subseteq, \owedge)$	Σ^0_1-hard
$\mathrm{TeamLTL}(\subseteq, \owedge, \mathsf{A})$	Σ^1_1-hard
$\mathrm{TeamLTL}(\sim)$	complete for third-order arithmetic

- 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 54

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 $\{j_1, j_2\}$, C_a^- goto $\{j_1, j_2\}$, 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 54

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

- 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 55 (Alur & Henzinger 1994, Schnoebelen 2010)

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.*

Undecidability results

Theorem 55 (Alur & Henzinger 1994, Schnoebelen 2010)

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 56 ([Vir+21])

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
- $(T[i, \infty], 0)$ encodes the value of counters of the ith configuration
  the value of C_a is the cardinality of the set $\{t \in T[i, \infty] \mid t[0](c_a) = 1\}$

Model checking for $\mathrm{TeamLTL}(\subseteq, \owedge)$ 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 $\mathrm{TeamLTL}(\subseteq, \owedge)$-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. $\square$

Idea of the encoding

Put $n := |I|$. A set T of traces using propositions $\{c_l, c_m, c_r, d, 0, \ldots, n-1\}$ encodes the sequence $(\vec{c}_j)_{j\in\mathbb{N}}$ of configurations, if for each $j \in \mathbb{N}$ and $\vec{c}_j = (i, v_l, v_m, v_r)$

- $t[j] \cap \{0, \ldots, n-1\} = \{i\}$, for all $t \in T$,
- $|\{t[j,\infty] \mid c_s \in t[j], t \in T\}| = v_s$, for each $s \in \{l, m, r\}$.

Hence, we use $T[j,\infty]$ to encode the configuration $\vec{c}_j$; the propositions $0, \ldots, n-1$ are used to encode the next instruction, and c_l, c_m, c_r, d are used to encode the values of the counters. The proposition d is a dummy proposition used to separate traces with identical postfixes with respect to c_l, c_m, and c_r.

Construction of the Kripke structure

The Kripke structure $\mathfrak{K}_I = (W, R, \eta, w_0)$ over the set of propositions $\{c_l, c_m, c_r, d, 0, \ldots, n-1\}$ is defined such that every possible sequence of configurations of M starting from $(0, 0, 0, 0)$ can be encoded by some team $(T, 0)$, where $T \subseteq \mathrm{Traces}(\mathfrak{K}_I)$.

Construction of the formula

The connective $\vee_{\mathrm{L}}$ is a shorthand for the condition:

$$(T, i) \models \phi \vee_{\mathrm{L}} \psi \text{ iff } \exists T_1, T_2 \text{ s.t. } T_1 \neq \emptyset,\; T_1 \cup T_2 = T, (T_1, i) \models \phi \text{ and } (T_2, i) \models \psi.$$

The disjunction $\vee_{\mathrm{L}}$ can be defined using $\subseteq$, $\vee$ (see e.g., [Hel+19, Lemma 3.4]).

Construction of the formula

The connective $\vee_{\mathrm{L}}$ is a shorthand for the condition:

$$(T, i) \models \phi \vee_{\mathrm{L}} \psi \text{ iff } \exists T_1, T_2 \text{ s.t. } T_1 \neq \emptyset,\ T_1 \cup T_2 = T, (T_1, i) \models \phi \text{ and } (T_2, i) \models \psi.$$

The disjunction $\vee_{\mathrm{L}}$ can be defined using $\subseteq$, $\vee$ (see e.g., [Hel+19, Lemma 3.4]).

The formula

$$\theta_{b-\mathrm{rec}} := \mathsf{GF}b$$

describes the b-recurrence condition of the computation.

Construction of the formula

The connective $\vee_{\mathrm{L}}$ is a shorthand for the condition:

$$(T, i) \models \phi \vee_{\mathrm{L}} \psi \text{ iff } \exists T_1, T_2 \text{ s.t. } T_1 \neq \emptyset,\ T_1 \cup T_2 = T, (T_1, i) \models \phi \text{ and } (T_2, i) \models \psi.$$

The disjunction $\vee_{\mathrm{L}}$ can be defined using $\subseteq$, $\vee$ (see e.g., [Hel+19, Lemma 3.4]).

The formula

$$\theta_{b-\mathrm{rec}} := \mathsf{GF}b$$

describes the b-recurrence condition of the computation.

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.

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

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

Expressing legality of computation

Define

$$\text{singleton} \coloneqq \mathsf{G} \bigwedge_{a \in \mathrm{PROP}} (a \otimes \neg a), \qquad \mathsf{c_s}\text{-decrease} \coloneqq c_s \vee (\neg c_s \wedge \mathsf{X} \neg c_s), \text{ for } s \in \{l, m, r\}.$$

Expressing legality of computation

Define

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

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

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

Expressing legality of computation

Define

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

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

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

For the instruction $i\colon$ if $C_s = 0$ goto j, else goto j', define

$$\theta_i := \big(\mathsf{X}(\neg c_s \wedge j) \oslash (\top \subseteq c_s \wedge \mathsf{X} j')\big) \wedge \mathsf{c_l}\text{-decrease} \wedge \mathsf{c_m}\text{-decrease} \wedge \mathsf{c_r}\text{-decrease}\,.$$

Expressing legality of computation

Define

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

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

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

For the instruction $i\colon$ if $C_s = 0$ goto j, else goto j', define

$$\theta_i := \big(\mathsf{X}(\neg c_s \wedge j) \varovee (\top \subseteq c_s \wedge \mathsf{X} j')\big) \wedge \mathsf{c_l}\text{-decrease} \wedge \mathsf{c_m}\text{-decrease} \wedge \mathsf{c_r}\text{-decrease}\,.$$

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

Conclusion of Lecture 5

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

[BRV01] Patrick Blackburn, Maarten de Rijke and Yde Venema. *Modal Logic*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2001. DOI: 10.1017/CBO9781107050884.

[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.

[Coo71] Stephen A. Cook. 'The Complexity of Theorem-Proving Procedures'. In: *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, May 3-5, 1971, Shaker Heights, Ohio, USA*. Ed. by Michael A. Harrison, Ranan B. Banerji and Jeffrey D. Ullman. ACM, 1971, pp. 151–158. DOI: 10.1145/800157.805047. URL: https://doi.org/10.1145/800157.805047.

[EFT94] Heinz-Dieter Ebbinghaus, Jörg Flum and Wolfgang Thomas. *Mathematical logic (2. ed.)* Undergraduate texts in mathematics. Springer, 1994.

[EJ99] E. Allen Emerson and Charanjit S. Jutla. 'The Complexity of Tree Automata and Logics of Programs'. In: *SIAM J. Comput.* 29.1 (1999), pp. 132–158.

[ES84] E. Allen Emerson and A. Prasad Sistla. 'Deciding Full Branching Time Logic'. In: *Inf. Control.* 61.3 (1984), pp. 175–201.

[FL79] Michael J. Fischer and Richard E. Ladner. 'Propositional Dynamic Logic of Regular Programs'. In: *J. Comput. Syst. Sci.* 18.2 (1979), pp. 194–211.

[GHR95] Raymond Greenlaw, H. James Hoover and Walter L. Ruzzo. *Limits to Parallel Computation: P-completeness Theory*. New York, NY, USA: Oxford University Press, Inc., 1995. ISBN: 0-19-508591-4.

[Gol77] L. M. Goldschlager. 'The monotone and planar circuit value problems are log-space complete for P'. In: *SIGACT News* 9 (1977), pp. 25–29.

[Gut+22] 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. URL: https://doi.org/10.1145/3531130.3533360.

[Han+18] Miika Hannula, Juha Kontinen, Jonni Virtema and Heribert Vollmer. 'Complexity of Propositional Logics in Team Semantic'. In: *ACM Trans. Comput. Log.* 19.1 (2018), 2:1–2:14.

[Han19] Miika Hannula. 'Validity and Entailment in Modal and Propositional Dependence Logics'. In: *Logical Methods in Computer Science* Volume 15, Issue 2 (Apr. 2019). DOI: 10.23638/LMCS-15(2:4)2019. URL: https://lmcs.episciences.org/5403.

[Hel+14] Lauri Hella, Kerkko Luosto, Katsuhiko Sano and Jonni Virtema. 'The Expressive Power of Modal Dependence Logic'. In: *Advances in Modal Logic*. College Publications, 2014, pp. 294–312.

[Hel+19] Lauri Hella, Antti Kuusisto, Arne Meier and Jonni Virtema. 'Model checking and validity in propositional and modal inclusion logics'. In: *J. Log. Comput.* 29.5 (2019), pp. 605–630.

[Hel+20] Lauri Hella, Antti Kuusisto, Arne Meier and Heribert Vollmer. 'Satisfiability of Modal Inclusion Logic: Lax and Strict Semantics'. In: *ACM Trans. Comput. Log.* 21.1 (2020), 7:1–7:18.

[HS15] Lauri Hella and Johanna Stumpf. 'The expressive power of modal logic with inclusion atoms'. In: *GandALF*. Vol. 193. EPTCS. 2015, pp. 129–143.

[KV85] Gabriel M. Kuper and Moshe Y. Vardi. 'On the Expressive Power of the Logical Data Model (Preliminary Report)'. In: *SIGMOD Conference*. ACM Press, 1985, pp. 180–187.

[Lev73] Leonid A. Levin. 'Universal sequential search problems'. In: *Problemy Peredachi Informatsii* 9.3 (1973).

[Loh12] Peter Lohmann. 'Computational Aspects of Dependence Logic'. PhD thesis. Leibniz Universität Hannover, 2012. arXiv: `1206.4564`. URL: `http://arxiv.org/abs/1206.4564`.

[LV19] Martin Lück and Miikka Vilander. 'On the Succinctness of Atoms of Dependency'. In: *Log. Methods Comput. Sci.* 15.3 (2019).

[Pap07] Christos H. Papadimitriou. *Computational complexity*. Academic Internet Publ., 2007.

[Pra80] V. R. Pratt. ‘A near-optimal method for reasoning about action’. In: *Journal of Computer and System Sciences* 20.2 (1980), pp. 231–254.

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

[Sch02] P. Schnoebelen. ‘The Complexity of Temporal Logic Model Checking’. In: *Advances in Modal Logic*. Vol. 4. 2002, pp. 393–436.

[Sip97] Michael Sipser. *Introduction to the theory of computation*. PWS Publishing Company, 1997.

[Var09] Moshe Y. Vardi. ‘From Philosophical to Industrial Logics’. In: *ICLA*. Vol. 5378. Lecture Notes in Computer Science. Springer, 2009, pp. 89–115.

[Vir+21] 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.

[Vir17] Jonni Virtema. 'Complexity of validity for propositional dependence logics'. In: *Inf. Comput.* 253 (2017), pp. 224–236.

[YV17] Fan Yang and Jouko Väänänen. 'Propositional team logics'. In: *Ann. Pure Appl. Log.* 168.7 (2017), pp. 1406–1441. DOI: 10.1016/J.APAL.2017.01.007. URL: https://doi.org/10.1016/j.apal.2017.01.007.