# Temporal Team Semantics Revisited

Jens Oliver Gutsfeld[1] Arne Meier[2] Christoph Ohrem[1] Jonni Virtema[2]

[1] Universität Münster, Germany
[2] Leibniz Universität Hannover, Germany

9.8.21 — LoDE'21

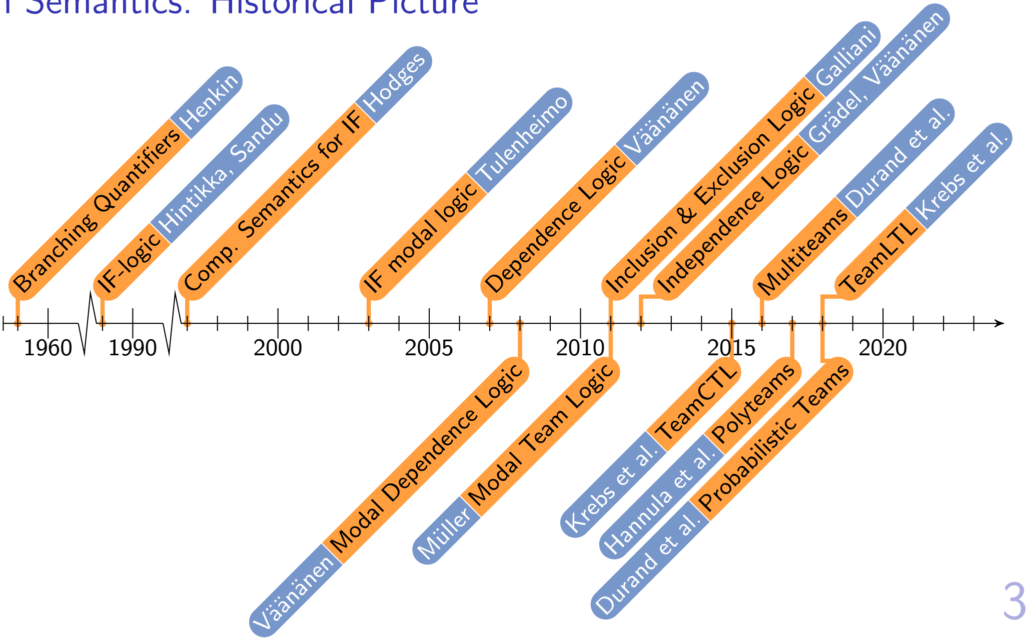
# Core of Team Semantics

- In most studied logics formulae are evaluated in a single state of affairs. E.g.,
  - a first-order assignment in first-order logic,
  - a propositional assignment in propositional logic,
  - a possible world of a Kripke structure in modal logic.

# Core of Team Semantics

- In most studied logics formulae are evaluated in a single state of affairs.
  E.g.,
  - a first-order assignment in first-order logic,
  - a propositional assignment in propositional logic,
  - a possible world of a Kripke structure in modal logic.
- In team semantics sets of states of affairs are considered.
  E.g.,
  - a set of first-order assignments in first-order logic,
  - a set of propositional assignments in propositional logic,
  - a set of possible worlds of a Kripke structure in modal logic.
- These sets of things are called teams.

# Team Semantics: Historical Picture



# Team semantics for temporal logics

- A trace over $\mathrm{AP}$ is an infinite sequence from $(2^{\mathrm{AP}})^{\omega}$.
- Trace can be seen to model an execution of a system over time.
- Important logics for trace properties are, e.g., LTL, CTL, $\mu$-calculus.
  - The system will terminate eventually.
  - Every request is eventually granted.
  - The system will terminate in bounded time.

# Team semantics for temporal logics

- A trace over $\mathrm{AP}$ is an infinite sequence from $(2^{\mathrm{AP}})^\omega$.
- Trace can be seen to model an execution of a system over time.
- Important logics for trace properties are, e.g., LTL, CTL, $\mu$-calculus.
  - The system will terminate eventually.
  - Every request is eventually granted.
  - The system will terminate in bounded time.
- A trace property is *a property of traces* (the set of satisfying traces) vs. a hyperproperty is *a property of sets of traces* (analogous to a set of teams).
- Logics for hyperproperties: HyperLTL, HyperCTL, TeamLTL, etc.
  - Termination in bounded time is in TeamLTL, but not in HyperLTL.
- Ongoing work on TeamLTL variants in Hannover, Helsinki, Münster, and Saarbrücken.

# LTL, HyperLTL, and TeamLTL

- In LTL the satisfying object is a trace.

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

- In HyperLTL the satisfying object is a set of traces and a trace assignment.

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

- In TeamLTL the satisfying object is a set of traces. We use team semantics.

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

$+$ atomic statements of dependence (dependence and inclusion atoms etc.)
$+$ additional connectives (Boolean disjunction, contradictory negation, etc.)

# Examples: HyperLTL vs. synchronous TeamLTL

- There is a timepoint (common for all traces) after which $a$ does not occur.
  Not expressible in HyperLTL, but is in HyperQPTL.

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

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

# Examples: HyperLTL vs. synchronous TeamLTL

- There is a timepoint (common for all traces) after which $a$ does not occur. Not expressible in HyperLTL, but is in HyperQPTL.

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

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

- Depending on an unknown input, execution traces either agree on $a$ or on $b$. Expressible in HyperLTL with three trace quantifiers:

$$\exists \pi_1 \, \exists \pi_2 \, \forall \pi \, \mathsf{G}(a_{\pi_1} \leftrightarrow a_\pi) \vee \mathsf{G}(b_{\pi_2} \leftrightarrow b_\pi).$$

  Expressible in synchronous TeamLTL: $\mathsf{G}(a \varovee \neg a) \vee \mathsf{G}(b \varovee \neg b)$.

# Motivation of the current work

- recent interest into temporal team semantics [KMVZ18, Lück20, VHFKY21]
- develop purely modal logics for hyperproperties
- investigate connections between HyperLTL variants and team semantics
- study different aspects of asynchronicity as most works on TeamLTL have concentrated on the synchronous semantics.

# Kripke structures and traces

A rooted Kripke structure is 4-tuple $(W, R, V, r)$, where

- $W$ is a (finite) set of states of the structure.
- the element $r \in W$ is the root of the structure.
- $R$ is a right-total binary relation on $W$ (i.e, $\forall x \in W \, \exists y \in W$ s.t. $xRy$).
- $V \colon W \to 2^{\mathsf{AP}}$ is an evaluation function.

A trace $t$ over $\mathrm{K}$ is an infinite sequence s.t $t[0] = r$ and $t[i]Rt[i+1]$, for $i \in \mathbb{N}$.
($t[i]$ is the $i$th element of the sequence $t$.)

# Time evaluation functions

## Definition

Given a (possibly infinite) set of traces $T$ over some common Kripke structure, a time evaluation function (tef for short) for $T$ is a function

$$\tau \colon \mathbb{N} \times T \to \mathbb{N}$$

that given a trace $t \in T$ and a value of a the global clock $i \in \mathbb{N}$ outputs the value $\tau(i, t)$ of the local clock of trace $t$ at global time $i$.

If $\tau$ is a tef and $k \in \mathbb{N}$ a natural number, then $\tau[k, \infty]$ is the $k$-shifted tef defined by putting $\tau[k, \infty](i, t) \coloneqq \tau(i + k, t)$, for everty $t \in T$ and $i \in \mathbb{N}$.

Note: there exists a notion of trajectory for hyperproperties [BCBFS21]

# Temporal teams

**Definition**

A temporal team is a tuple $(T, \tau)$, where $T$ is a set of traces over some common Kripke structure and $\tau$ is a time evaluation function for $T$. A pair $(T, \tau)$ is called a stuttering temporal team, if $(S, \tau)$ is a temporal team for some $T \subseteq S$.

# Temporal team semantics

## Definition

Let $(T, \tau)$ be a stuttering temporal team over a Kripke structure $(W, R, V, r)$.

$$
\begin{array}{llllll}
(T,\tau) \models p & \text{iff} & \forall t \in T : p \in V(t[0]) & (T,\tau) \models \neg p & \text{iff} & \forall t \in T : p \notin V(t[0]) \\
(T,\tau) \models \phi \wedge \psi & \text{iff} & (T,\tau) \models \phi \text{ and } (T,\tau) \models \psi & (T,i) \models \mathsf{X}\varphi & \text{iff} & (T,\tau[1,\infty]) \models \varphi
\end{array}
$$

$$
\begin{array}{lll}
(T,\tau) \models \phi \vee \psi & \text{iff} & (T_1,\tau) \models \phi \text{ and } (T_2,\tau) \models \psi, \text{ for some } T_1, T_2 \text{ s.t. } T_1 \cup T_2 = T \\
(T,\tau) \models \phi \mathsf{U} \psi & \text{iff} & \exists k \in \mathbb{N} \text{ s.t. } (T,\tau[k,\infty]) \models \psi \text{ and } \forall m : 0 \leq m < k \Rightarrow (T,\tau[m,\infty]) \models \phi \\
(T,\tau) \models \phi \mathsf{W} \psi & \text{iff} & \forall k \in \mathbb{N} : (T,\tau[k,\infty]) \models \phi \text{ or } \exists m \text{ s.t. } m \leq k \text{ and } (T,\tau[m,\infty]) \models \psi
\end{array}
$$

Note: If $\tau$ is the synchronous time evaluation function (i.e., $\forall t \forall i : \tau(t,i) = i$), then the above is exactly the semantics for synchronous TeamLTL as defined in [KMVZ18].

# Properties of tefs

* marks optional properties

Strict Monotonicity: $\forall i : \tau(i) < \tau(i+1)$ (wrt. canonical order of tuples)

Stepwise: $\forall i \, \forall t : \tau(i+1, t) \in \{\tau(i,t), \tau(i,t)+1\}$.
Whenever a local clock ticks it ticks exactly one step.
Important to differentiate neXt operator from Future.

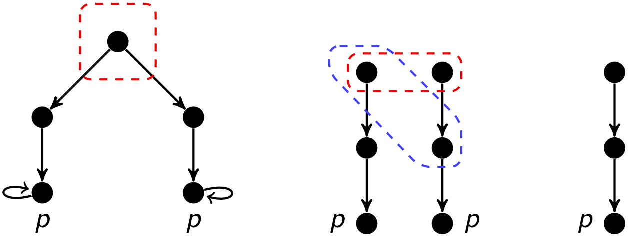
*Fairness: $\forall i \, \forall t \exists j : \tau(j,t) \geq i$.

*Non-Parallelism: $\forall i : i = \sum_t \tau(i,t)$

*Synchronousity: $\tau(i,t) = \tau(i,t')$ for all $i, t, t'$.

# An initial example

The formula $XX\neg p$ is not downward closed due to strict monotonicity!



# Quantification of tefs

### Definition

Fix a set AP of atomic propositions. The set of formulae of TeamLTL (over AP) is generated by the following grammar:

$$\varphi ::= p \mid \neg p \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \mathsf{X}\varphi \mid \varphi \mathsf{U} \varphi \mid \varphi \mathsf{W} \varphi$$

where $p \in \mathrm{AP}$.
The logical constants $\top, \bot$ and connectives $\rightarrow, \leftrightarrow$ are defined as usual (e.g., $\bot := p \wedge \neg p$), and $\mathsf{F}\phi := \top \mathsf{U} \phi$ and $\mathsf{G}\phi := \phi \mathsf{W} \bot$.

# Quantification of tefs

## Definition

Fix a set AP of atomic propositions. The set of formulae of TeamCTL* (over AP) is generated by the following grammar:

$$\varphi ::= p \mid \neg p \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \mathsf{X}\varphi \mid \varphi \mathsf{U} \varphi \mid \varphi \mathsf{W} \varphi \mid \exists \mid \forall$$

where $p \in \mathrm{AP}$ and $\exists, \forall$ are tef quantifiers.
The logical constants $\top, \bot$ and connectives $\rightarrow, \leftrightarrow$ are defined as usual (e.g., $\bot := p \wedge \neg p$), and $\mathsf{F}\phi := \top \mathsf{U} \phi$ and $\mathsf{G}\phi := \phi \mathsf{W} \bot$.

Note: The naming of above logics are work in progress.

# Quantification of tefs

**Definition**

Fix a set AP of atomic propositions. The set of formulae of TeamCTL (over AP) is generated by the following grammar:

$$\varphi ::= p \mid \neg p \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \exists \mathsf{X} \varphi \mid \exists \varphi \mathsf{U} \varphi \mid \exists \varphi \mathsf{W} \varphi \mid \forall \mathsf{X} \varphi \mid \forall \varphi \mathsf{U} \varphi \mid \forall \varphi \mathsf{W} \varphi$$

where $p \in \mathrm{AP}$.
The logical constants $\top, \bot$ and connectives $\rightarrow, \leftrightarrow$ are defined as usual (e.g., $\bot := p \wedge \neg p$), and $\mathsf{F}\phi := \top \mathsf{U} \phi$ and $\mathsf{G}\phi := \phi \mathsf{W} \bot$.

Note: The naming of above logics are work in progress.

# TeamCTL can simulate synch-TeamLTL: the existential fragment

$$\psi_{\text{synch}} \coloneqq (p \wedge \mathsf{X}_{\exists}\neg p) \ovee (\neg p \wedge \mathsf{X}_{\exists} p),$$
$$\psi'_{\text{synch}} \coloneqq (p \wedge \mathsf{X}_{\exists}\neg p) \vee (\neg p \wedge \mathsf{X}_{\exists} p),$$
$$\psi''_{\text{synch}} \coloneqq \big(p \wedge \mathsf{X}_{\forall}(\neg p \vee p \subseteq \neg p)\big) \vee \big(\neg p \wedge \mathsf{X}_{\forall}(p \vee p \subseteq \neg p)\big),$$

Then: $\mathsf{G}_{\exists}\psi_{\text{synch}}$ states that on every trace $p$ flips from each time step to next one.
without $\ovee$: $p \wedge \mathsf{G}_{\forall}\psi'_{\text{synch}}$ and $p \wedge \mathsf{G}_{\forall}\psi''_{\text{synch}}$
with fairness: $p \wedge \mathsf{G}_{\exists}\psi'_{\text{synch}}$
note: formulas needed for SAT, for MC encode alternation of $p$ into model.

# TeamCTL can simulate synch-TeamLTL: the existential fragment II

Now turn towards the formulas for the respective operators:

$$
\begin{aligned}
(\mathsf{F}\varphi)^* &\coloneqq [\mathsf{dep}(p)\mathsf{U}_\exists\varphi \land \mathsf{dep}(p)]\\
(\mathsf{G}\varphi)^* &\coloneqq [\varphi \land \mathsf{dep}(p)\mathsf{W}_\exists\bot]\\
(\mathsf{X}\varphi)^* &\coloneqq \mathsf{X}_\exists\big(\mathsf{dep}(p) \land \varphi\big)\\
(\varphi\mathsf{U}\theta)^* &\coloneqq [\varphi \land \mathsf{dep}(p)\mathsf{U}_\exists\theta \land \mathsf{dep}(p)]\\
(\varphi\mathsf{W}\theta)^* &\coloneqq [\varphi \land \mathsf{dep}(p)\mathsf{W}_\exists\theta \land \mathsf{dep}(p)]
\end{aligned}
$$

The translation then is

$$\varphi \mapsto (\varphi)^* \land \theta,$$

where $\theta$ is any of the formulas $\mathsf{G}_\exists\psi_{\mathsf{synch}}$, $p \land \mathsf{G}_\forall\psi'_{\mathsf{synch}}$, or $p \land \mathsf{G}_\exists\psi'_{\mathsf{synch}}$ (if fairness for time evaluation is stipulated), and where for Boolean connectives the translation is the identity.

# 2-Counter-Machines

## Definition

A non-deterministic 2-counter machine $M$ consists of a list $I$ of $n$ instructions that manipulate two counters $C_l$ 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 $a \in \{l, r\}$, $0 \leq j_1, j_2 < n$.

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

## Theorem (Alur & Henzinger 1994 )

*Deciding whether a given non-deterministic 2-counter machine has a b-recurring computation for a given b is $\Sigma_1^1$-complete.*

# TeamCTL($\varovee$) is highly undecidable

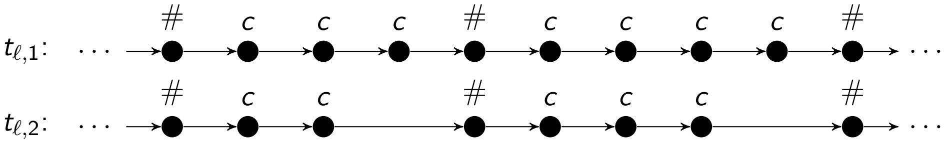
**Theorem**

*Model checking for* TeamCTL($\varovee$) *is* $\Sigma_1^1$*-hard.*

Proof Idea: reduce existence of $b$-recurring computation of given 2-counter machine $M$ and instruction label $b$ to model checking problem of TeamCTL($\varovee$).

# Using traces to simulate the counters

- use two traces $t_{\ell,1}, t_{\ell,2}$ for counter $C_\ell$ and two traces $t_{r,1}, t_{r,2}$ for counter $C_r$
- each $t \in \{t_{\ell,1}, t_{\ell,2}, t_{r,1}, t_{r,2}\}$ has $p_t$ that is globally true in each state of trace
- trace-pairs simulate incrementing, resp., decrementing value of respective counter
- counter-value $n \in \mathbb{N}$ simulated via sequence of $n$ states containing $c$.
- between such sequence use separation symbol $\#$



# Use TeamCTL-formulas to express the details

Excerpt of details:

- label $b$ reoccurs infinitely often: $\mathsf{G}_\forall \mathsf{F}_\exists b$
- Increment $C_\ell$: $\neg \# \mathsf{U}_{\mathrm{synch}} \Big( c \wedge (p_{t_s,2} \wedge \mathsf{X}_\exists \neg c) \vee (p_{t_s,1} \wedge \mathsf{X}_\exists c) \Big)$
- Decrement $C_\ell$: $\neg \# \mathsf{U}_{\mathrm{synch}} \Big( c \wedge (p_{t_s,2} \wedge \mathsf{X}_\exists c) \vee (p_{t_s,1} \wedge \mathsf{X}_\exists \neg c) \Big)$
- Stay $C_\ell$: $\neg \# \mathsf{U}_{\mathrm{synch}} (c \wedge \mathsf{X}_\exists \neg c)$
- instruction formulas:

  $i$: $C_s^+$ goto $\{j, j'\}$: $\big(c_s\text{-inc} \wedge (p_{t_s,1} \vee p_{t_s,2})\big) \vee \big(c_{\bar{s}}\text{-stay} \wedge (p_{t_{\bar{s}},1} \vee p_{t_{\bar{s}},2})\big)$

  $i$: $C_s^-$ goto $\{j, j'\}$: $\big(c_s\text{-dec} \wedge (p_{t_s,1} \vee p_{t_s,2})\big) \vee \big(c_{\bar{s}}\text{-stay} \wedge (p_{t_{\bar{s}},1} \vee p_{t_{\bar{s}},2})\big)$

  $i$: if $C_s = 0$ goto $j$, else goto $j'$:

$$\Big(p_{t_s,2} \wedge \Big( (\# \wedge \mathsf{X}_\exists(\neg c \wedge j)) \otimes (\# \wedge \mathsf{X}_\exists(c \wedge j')) \Big) \vee p_{t_s,1} \vee p_{t_{\bar{s}},1} \vee p_{t_{\bar{s}},2}$$

# Summary

- General framework for temporal team semantics
- We can combine asynchronous and synchronous tefs
- We can embed synchronous TeamLTL
- highly undecidable model-checking problem

# Summary

- General framework for temporal team semantics
- We can combine asynchronous and synchronous tefs
- We can embed synchronous TeamLTL
- highly undecidable model-checking problem

**Current and future directions**

- Indentification of decidable fragments and variants
- Consider tefs also as inputs given in some finite way.

# Summary

- General framework for temporal team semantics
- We can combine asynchronous and synchronous tefs
- We can embed synchronous TeamLTL
- highly undecidable model-checking problem

**Current and future directions**

- Indentification of decidable fragments and variants
- Consider tefs also as inputs given in some finite way.

**Thank you!**

# Advert

I will start a lecturer position in the University of Sheffield (UK) in September.

**Tentative Open Positions:**

- Postdoc position for 26 months. Probabilistic team semantics, descriptive complexity of BSS-computation, connections to quantum information theory. International call, ASAP. `www.virtema.fi/dfg`
- Fully funded PhD position for 3.5 years in Sheffield. UK students only. Starting in the academic year 2021 or 2022.

More details: `jonni.virtema@gmail.com`