Verifying Hyperproperties with TLA

Fred B. Schneider
Samuel B Eckert Professor of Computer Science

Department of Computer Science
Cornell University
Ithaca, New York 14853
U.S.A.

Joint work with Leslie Lamport

Appears in:
Proceedings 4th IEEE Computer Security Foundations Symposium.
June 2021

Overview

- Hyperproperties
  - What and Why
- TLA
  - Important characteristics
- TLA verification of hyperproperties

Hyperproperties

behavior: an infinite sequence of states

$$\sigma = s_0\, s_1\, s_2\ \dots\ s_i\ \dots$$

Property: A predicate on individual behaviors.

- Any sequential, concurrent, or distributed program (!)
- Partial correctness, total correctness
- Mutual exclusion
- Termination / Eventual service

Hyperproperty: A predicate on sets of behaviors.

- Information flow
- Memory consistency

Verification of Hyperproperties?

- Led to “new” methods being created.
  - Logix x + more stuff = Logic hyper-x

- But new methods are not necessary!
  - What attributes of an existing method are required?
  - Why.

Properties and $\hat{P}$redicates

Property: set P of behaviors defined by predicates $\hat{P}$ on behaviors σ:

$$\sigma \models \hat{P} \stackrel{\text{def}}{=} \hat{P} \text{ is true on } \sigma$$
$$= \sigma \in P$$

Program: predicate $\hat{S}$ on behaviors that defines a set S of behaviors

$$S \models \hat{P} = S \subseteq P$$
$$= (\forall \sigma : \sigma \in S \Rightarrow \sigma \in P)$$
$$= \left(\forall \sigma : \sigma \models \hat{S} \Rightarrow \sigma \models \hat{P}\right)$$
$$= \left(\forall \sigma : \sigma \models \left(\hat{S} \Rightarrow \hat{P}\right)\right)$$
$$= \models \left(\hat{S} \Rightarrow \hat{P}\right)$$

TLA is a logic where programs S are easily expressed as formulas $\hat{S}$.

Universal Domains for States (rqmnt)

"$\models \left(\hat{S} \Rightarrow \hat{P}\right)$" means "*true* in <u>all</u> interpretations σ"!

σ: $s_0\, s_1 \ldots s_i \ldots$ What variables does s_i map?

Expect:

$$\frac{\models \hat{P}, \ \models \hat{Q}}{\models \hat{P} \wedge \hat{Q}}$$

Soundness then requires:

States in a behavior σ must map **all** variables to values.

… including variables not in $\hat{P}$ and not in $\hat{Q}$

Stuttering (rqmnt)

Example: Clock specifications (seconds shouldn't matter)

$\widehat{HMS}$ behaviors are increasing: hrs h, mins m, secs s;
$\widehat{HM}$ behaviors are increasing: hrs h, mins m.

$\models \widehat{HMS} \Rightarrow \widehat{HM}$?

<3:59:50> … <3:59:59> <4:00:00> … $\models \widehat{HMS}$

<3:59> <4:00> … $\models \widehat{HM}$

<3:59:50> … <3:59:59> <4:00:00> … $\neg\models \widehat{HM}$

$\neg\models \widehat{HMS} \Rightarrow \widehat{HM}$!

Conclusion: Predicates must be **stuttering insensitive** or else they constrain unnamed variables. *Specifications should constrain a system but not the whole universe!*

Toward verification of hyperproperties:

Hyperproperties as Predicates

A hyperproperty is defined by a predicate on properties P.

A **finitary hyperproperty** $\mathcal{H}(P)$ is always equivalent to

$$\forall/\exists \sigma_1 \in P\text{:} \ \ldots \ \forall/\exists \sigma_k \in P\text{:} \ \ \hat{J}(\sigma_1, \ldots, \sigma_k)$$

where $\hat{J}(\cdot)$ does not depend on P.

Translate Sets to Predicates 1/2

$$\forall/\exists\sigma_1 \in P: \ldots \forall/\exists\sigma_k \in P: \ \hat{J}(\sigma_1, \ldots, \sigma_k)$$

Translate: Set membership to predicate satisfaction

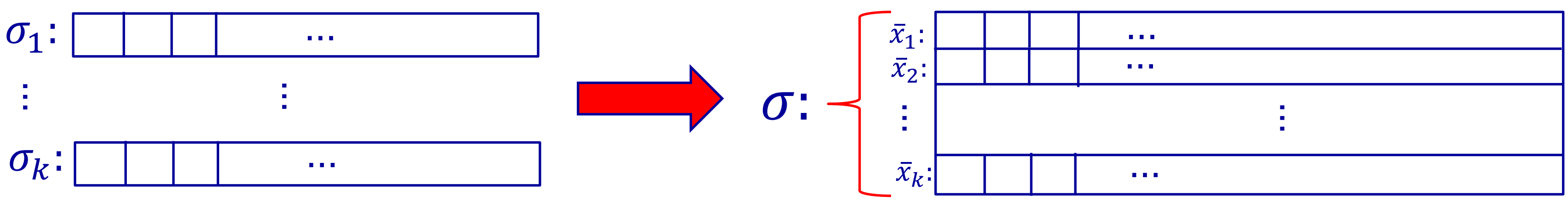
- $\forall\sigma \in P: \ldots$ into $\forall\sigma: \hat{P}(\sigma) \Rightarrow \ \ldots$
- $\exists\sigma \in P: \ldots$ into $\exists\sigma: \hat{P}(\sigma) \wedge \ \ldots$

Translate Sets to Predicates 2/2

Translate: Predicates on <u>behaviors</u> to Temporal Logic formulas on <u>variables</u>

- $\hat{J}(\ldots, \sigma_i, \ldots)$ into $\hat{J}(\ldots, \bar{x}_i, \ldots)$

where $\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_k$ are disjoint lists. [*Cf Self-Composition]*



Temporal Logic inference or model checking does the rest.

We have: Reduced hyperproperty verif to property verif!

∀∃-Hyperproperties in TLA

A subclass of finitary hyperproperties:

$$\hat{P}(\bar{x}_1) \wedge \cdots \wedge \hat{P}(\bar{x}_j) \wedge \hat{K}(\bar{x}_1, \ldots, \bar{x}_j)$$
$$\Rightarrow (\exists \bar{x}_{j+1} \ldots, \bar{x}_k \colon \; \hat{P}(\bar{x}_{j+1}) \wedge \cdots \wedge \hat{P}(\bar{x}_k)$$
$$\wedge \; \hat{L}(\bar{x}_1, \ldots, \bar{x}_k))$$

- A class of formulas TLA^+ model checker handles.
- Class is expressive enough to handle all hyperproperties we have encountered in literature.

Why SI: GNI case study 1/4

Generalized Non-interference (GNI): For any behaviors σ_1 and σ_2 in P, there is a behavior σ_3 exhibiting the public events of σ_1 and the secret events of σ_2.

$$\widehat{P}(\bar{x}_1) \wedge \widehat{P}(\bar{x}_2) \;\Rightarrow\; (\exists \bar{x}_3 \colon \widehat{P}(\bar{x}_3) \wedge \widehat{L}(\bar{x}_1, \bar{x}_2, \bar{x}_3))$$

$$\widehat{L}(\bar{x}_1, \bar{x}_2, \bar{x}_3) \;\stackrel{\text{def}}{=}\; \Box(\, pub(\bar{x}_3) = pub(\bar{x}_1) \wedge \sec(\bar{x}_3) = \sec(\bar{x}_2))$$

Why SI: GNI case study 2/4

$$\hat{P}(\bar{x}_1) \land \hat{P}(\bar{x}_2) \Rightarrow (\exists \bar{x}_3 \colon \hat{P}(\bar{x}_3) \land \hat{L}(\bar{x}_1, \bar{x}_2, \bar{x}_3))$$

$$\hat{L}(\bar{x}_1, \bar{x}_2, \bar{x}_3) \stackrel{\text{def}}{=} \Box(\, pub(\bar{x}_3) = pub(\bar{x}_1) \land sec(\bar{x}_3) = sec(\bar{x}_2))$$

Example: $\hat{P}(\bar{x}_1)$: steps $ps(\bar{x}_1)$ alternates with $ss(\bar{x}_1)$, where:

- $ps(\bar{x}_1)$ step updates $pub(\bar{x}_1)$ but not $sec(\bar{x}_1)$
- $ss(\bar{x}_1)$ step updates $sec(\bar{x}_1)$ but not $pub(\bar{x}_1)$

Why SI: GNI case study   3/4

$$\hat{P}(\bar{x}_1) \wedge \hat{P}(\bar{x}_2) \;\Rightarrow\; (\exists \bar{x}_3\colon \hat{P}(\bar{x}_3) \wedge \hat{L}(\bar{x}_1, \bar{x}_2, \bar{x}_3))$$

$$LL(\bar{x}_1, \bar{x}_2, \bar{x}_3) \;\stackrel{\text{def}}{=}\; \Box(\, pub(\bar{x}_3) = pub(\bar{x}_1) \wedge \mathrm{sec}(\bar{x}_3) = \mathrm{sec}(\bar{x}_2))$$

$\bar{x}_1$:	s_1	$ps(\bar{x}_1)$	s_2	$ss(\bar{x}_1)$	s_3	$ps(\bar{x}_1)$	s_4	...
$\bar{x}_2$:	t_1	$ps(\bar{x}_2)$	t_2		t_3	$ss(\bar{x}_2)$	t_4	...
$\bar{x}_3$:	u_1	$ps(\bar{x}_3)$	u_2		u_3	? ? ?	u_4	...

Stuttering Insensitivity (SI)

Behaviors σ and τ are **stuttering equivalent** if deleting repeated values from each produces identical sequences.

- Define $\sigma \vDash (f \sim g)$ iff $\sigma|_f$ and $\sigma|_g$ are stuttering equivalent, where **projection** $\sigma|_f$ is sequence of values σ gives to state function f.

TLA is a linear-time temporal logic where all formulas are SI.

- $\hat{S} \Rightarrow \hat{P}$ can mean $\hat{S}$ satisfies/implements $\hat{P}$
- Can "form" a behavior for execution that combines executions described by behaviors σ_1 and σ_2 (needed for some hyperproperties).

Why SI: GNI case study 4/4

$$\hat{P}(\bar{x}_1) \wedge \hat{P}(\bar{x}_2) \Rightarrow$$
$$(\exists \bar{x}_3, \bar{y}_1, \bar{y}_2\colon\ \ \bar{y}_1 \sim \bar{x}_1 \ \wedge\ \bar{y}_2 \sim \bar{x}_2 \ \wedge\ \hat{P}(\bar{x}_3)$$
$$\wedge\ \hat{L}(\bar{y}_1, \bar{y}_2, \bar{x}_3))$$

$$LL(\bar{x}_1, \bar{x}_2, \bar{x}_3) \ \stackrel{\text{def}}{=}\ \square(\, pub(\bar{x}_3) = pub(\bar{x}_1) \wedge sec(\bar{x}_3) = sec(\bar{x}_2))$$

- $\bar{y}_1 \sim \bar{x}_1$, $\bar{y}_2 \sim \bar{x}_2$ accounts for SI behaviors in $\hat{P}(\cdot)$.

∀∃-Hyperproperties Examples

- **Generalized Non-interference**: For any behaviors σ_1 and σ_2 in P, there is a behavior σ_3 exhibiting the public events of σ_1 and the secret events of σ_2.
- **Observational non-determinism**. Two system behaviors with same initial public state are public-stuttering equivalent.
- **Non-interference**. Deleting secret commands has no effect on public outputs.
- **Possibilistic non-interference**. If σ_1 and σ_2 have the same initial public values *then* the exists a behavior σ_3 with the same initial state as σ_2 and the same public values as σ_1 throughout.

Summary

- Hyperproperties provide needed expressiveness for security and concurrency.
- Existing logics + self composition works if:
  - States map all variables.
    - Already needed for ordinary compositionality
  - Behaviors are stuttering insensitive.
    - Already needed for "implements" to be implication ($\Rightarrow$)
- TLA+ is such a logic, used in industry and with a model checker for support.

Reading

- M. Clarkson and F.B. Schneider. Hyperproperties. *Journal of Computer Security 18* (2010).
- L. Lamport. The temporal logic of actions. *ACM Transactions on Programming Languages and Systems*, 16(3), May 1994, 872--923.
- L. Lamport and F.B. Schneider. Verifying hyperproperties with TLA. *34th IEEE Computer Security Foundations Symposium*. (Virtual Conference. June 2021), 1--16. Distinguished paper award.