# A Framework for Inherent Vacuity

Dana Fisman, Orna Kupferman,
Sarai Sheinvald-Faragy, Moshe Vardi

# What is Vacuity?

In Model Checking, we are given

a formula **φ** and a model **M** and check

$$\mathbf{M} \overset{?}{\models} \boldsymbol{\varphi}$$

**NO**: a counterexample is returned

**YES** … is this enough?

$$\mathbf{M} \models \mathbf{G}(\textit{req} \rightarrow \mathbf{F}\ \textit{grant})$$

***req* never occurs**

# What is Vacuity?

Model

LTL Formula

$M \models \varphi = [\ldots\ \psi\ \ldots]$

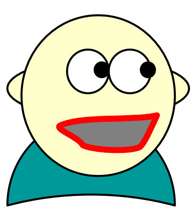
Subformula

$\psi$ does not affect $\varphi$ in **M**:

For every labeling of a path $\pi$ by $x$, $\varphi[\psi \leftarrow x]$ is satisfied

$M \models \forall x. \varphi\ [\psi \leftarrow x]$

The $\psi$-strengthening of $\varphi$



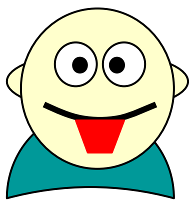
**M** satisfies $\varphi$ vacuously

# Inherent Vacuity

Sometimes, the problem is not in the model,
but in the **formula**.

Some formulas will be satisfied vacuously **in every model**.

<u>Examples</u>:
***true***



$$(\mathit{message} \rightarrow \mathbf{F}\ \mathit{button\_on}) \wedge (\neg \mathit{message} \rightarrow \mathbf{F}\ \mathit{button\_on})$$

We seek criteria that would help detect formulas that are satisfied vacuously, **regardless of the model**.

$$\mathbf{M} \models \varphi = [\ldots\ \psi\ \ldots]$$

# Inherent Vacuity - Motivation

[Chockler & Shtrichman 08]

**Vacuity without design**:
Testing the specification **before** model checking takes place

$$\varphi_1 \wedge \varphi_2 \ldots \wedge \varphi_{17} \; = \; \begin{array}{|c|}\hline \varphi_1 \\ \hline \varphi_2 \\ \hline \varphi_3 \\ \hline \varphi_4 \\ \hline \varphi_5 \\ \hline . \\ \hline . \\ \hline \varphi_{17} \\ \hline \end{array} \; \Rightarrow \; \varphi_2 \wedge \ldots \wedge \varphi_9$$

$$\varphi_1 \wedge \varphi_2 \ldots \wedge \varphi_{17} \rightarrow \varphi_1 \wedge \ldots \wedge \varphi_{17}$$

the cause of vacuity

- Exponentially cheaper than vacuity testing.
- Saves redundant testing and improvement
  (it's not the model's fault).

# Inherent Vacuity - Motivation

**<u>Causes for inherent vacuity</u>**

- Combination of non-vacuous formulas by different specifiers.

$\varphi 1$ $\varphi 2$ … $\varphi 2$

$\varphi_1 \wedge \varphi_2 \ldots \wedge \varphi_{17}$

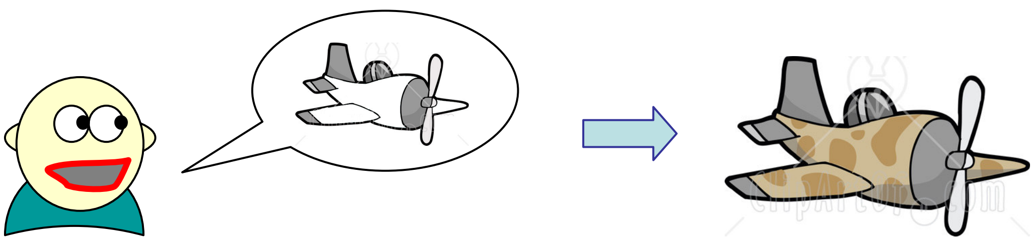
- Packing complicated formulas with common properties into a single property, common when using SVA/PSL

Prosyd: There is a need to formalize vacuity of specifications

# Inherent Vacuity - Motivation

**Property-based design**

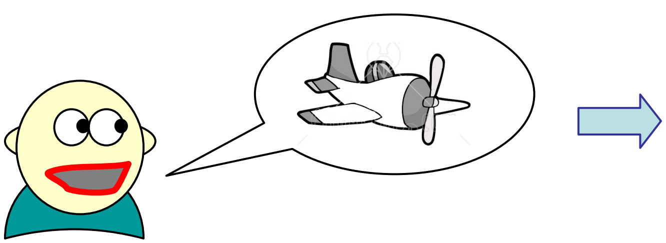
Classically, the model is built
and then tested for a desired specification.



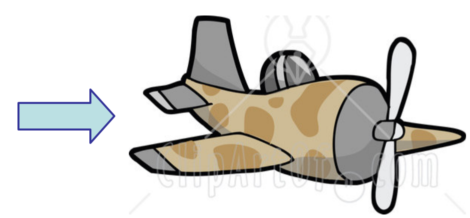
$\models \quad \varphi_1 \wedge \varphi_2 \ldots \wedge \neg F\ \textit{crash}$

In property-based design, the model is produced **from the specification**.

In **synthesis**, the model is produced automatically.



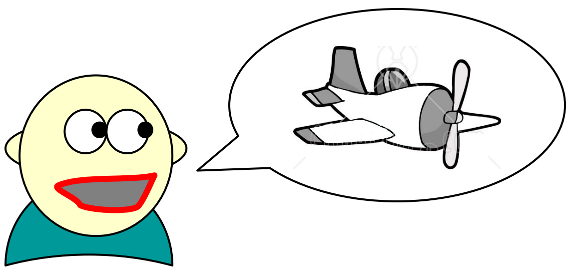
$\varphi_1 \wedge \varphi_2 \ldots \wedge \neg F\ \textit{crash}$



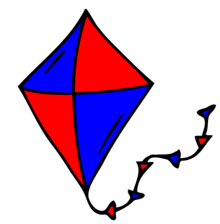
# Inherent Vacuity - Motivation

**Property-based design**

The difficulty is shifted from building a correct model to writing a correct specification.



**$\varphi_1 \wedge \varphi_2 \ldots \neg F$** *crash*



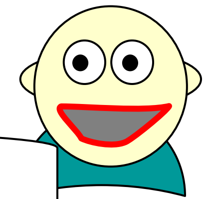
A wrong specification produces a wrong model

Vacuity of the specification may lead to models that lack desired properties or have too many.

Vacuity testing leads to better, simpler and more accurate specifications.

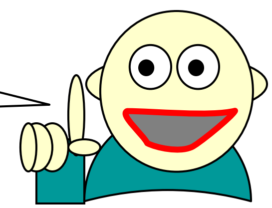
# Our Goal and Contribution

- Formally define inherent vacuity



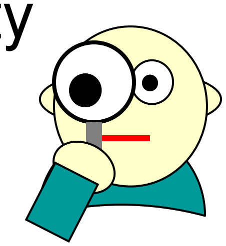
We study two natural approaches and prove equivalence

- Provide a **Framework** for inherent vacuity, covering the many different definitions for vacuity and suitable for all applications of vacuity and possibly more



The approaches are extended
and are equivalent all over the framework!

- Study the problem of deciding inherent vacuity and provide algorithms



# First Definition of Inherent Vacuity

$\varphi$ is **inherently vacuous by mutation** if there exists a subformula $\psi$ such that

$$\varphi \equiv \forall x.\varphi[\psi \leftarrow x]$$

witness for inherent vacuity

**Example:**

$\varphi = \mathbf{F}(\mathit{grant} \vee \mathit{fail}) \vee \mathbf{X}\, \mathit{fail}$

intuitively:

$\mathbf{M} \models \mathbf{F}(\mathit{grant} \vee \mathit{fail}) \rightarrow \mathbf{M} \models \mathbf{F}(\mathit{grant} \vee \mathit{fail}) \vee \mathbf{X}\, \mathit{fail}$

$\mathbf{M} \models \mathbf{F}(\mathit{grant} \vee \mathit{fail}) \vee \mathbf{X}\, \mathit{fail} \rightarrow \mathbf{M} \models \mathbf{F}(\mathit{grant} \vee \mathit{fail})$

$\mathbf{X}\, \mathit{fail} \rightarrow \mathbf{F}(\mathit{grant} \vee \mathit{fail})$

$\Rightarrow \quad \varphi = \mathbf{F}(\mathit{grant} \vee \mathit{fail})$

# First Definition of Inherent Vacuity

**φ** is **inherently vacuous by mutation** if there exists a subformula **ψ** such that

$$\varphi \equiv \forall x.\varphi[\psi \leftarrow x]$$

witness for inherent vacuity

**Example:**

**φ** = **F**(*grant* ∨ *fail*) ∨ **X** *fail*

**X***fail*-strengthening

≡ ∀x.**F**(*grant* ∨ *fail*) ∨ x ≡ **F**(*grant* ∨ *fail*)

# Second Definition of Inherent Vacuity

$\varphi$ is **inherently vacuous by model** if for every Kripke structure **K**,
if $K \models \varphi$ then **K** satisfies $\varphi$ vacuously.

The definition does not restrict attention to a specific subformula

… nevertheless…

**Theorem:** $\varphi$ is **inherently vacuous by model** iff
$\varphi$ is **inherently vacuous by mutation**

**Theorem:** $\boldsymbol{\varphi}$ is **inherently vacuous by mutation**

$\Downarrow$

$\boldsymbol{\varphi}$ is **inherently vacuous by model**

**Proof:**

$\boldsymbol{\varphi}$ is **inherently vacuous by mutation**

$\Downarrow$

$$\boldsymbol{\varphi} \equiv \forall x.\boldsymbol{\varphi}[\psi \leftarrow x]$$

$\Downarrow$

For every Kripke structure **K**,

if $\mathbf{K} \models \boldsymbol{\varphi}$ then $\mathbf{K} \models \forall x.\boldsymbol{\varphi}[\psi \leftarrow x]$

$\Downarrow$

$\boldsymbol{\varphi}$ is **inherently vacuous by model**

**Theorem**: $\boldsymbol{\varphi}$ is **inherently vacuous by mutation**

$\boldsymbol{\varphi}$ is **inherently vacuous by model**

**Proof** :

Assume $\boldsymbol{\varphi}[\psi_1, \psi_2, \ldots, \psi_t]$ is **not inherently vacuous by mutation**

$$\boldsymbol{\varphi} \neq \forall x.\boldsymbol{\varphi}[\psi_1 \leftarrow x]$$
$$\boldsymbol{\varphi} \neq \forall x.\boldsymbol{\varphi}[\psi_2 \leftarrow x]$$
$$\ldots$$
$$\boldsymbol{\varphi} \neq \forall x.\boldsymbol{\varphi}[\psi_t \leftarrow x]$$

$$\exists K_1 \text{ s.t. } K_1 \models \boldsymbol{\varphi} \quad \text{but} \quad K_1 \not\models \forall x.\boldsymbol{\varphi}[\psi_1 \leftarrow x]$$
$$\exists K_2 \text{ s.t. } K_2 \models \boldsymbol{\varphi} \quad \text{but} \quad K_2 \not\models \forall x.\boldsymbol{\varphi}[\psi_2 \leftarrow x]$$
$$\ldots$$
$$\exists K_t \text{ s.t. } K_t \models \boldsymbol{\varphi} \quad \text{but} \quad K_t \not\models \forall x.\boldsymbol{\varphi}[\psi_t \leftarrow x]$$

$$K_1 \cup K_2 \cup \ldots \cup K_t \models \varphi$$

For some $j$

$\boldsymbol{\varphi}[\psi_1, \psi_2, \ldots, \psi_t]$ is inherently vacuous by model

$$K_1 \cup K_2 \cup \ldots \cup K_t \models \forall x.\boldsymbol{\varphi}[\psi_j \leftarrow x]$$

$$K_j \models \forall x.\boldsymbol{\varphi}[\psi_j \leftarrow x]$$

**Contradiction!!!**

$\blacksquare$

**Theorem:**

Deciding whether $\boldsymbol{\varphi} \equiv \forall x.\boldsymbol{\varphi}[\boldsymbol{\psi} \leftarrow x]$ is PSPACE-Complete

**Proof:**

**upper bound:**

$\forall x.\boldsymbol{\varphi}[\boldsymbol{\psi} \leftarrow x] \rightarrow \boldsymbol{\varphi}$ always holds.

To check $\boldsymbol{\varphi} \rightarrow \forall x.\boldsymbol{\varphi}[\boldsymbol{\psi} \leftarrow x]$:

check the satisfiability of $\boldsymbol{\varphi} \wedge \exists x.\neg\boldsymbol{\varphi}[\boldsymbol{\psi} \leftarrow x]$:

$\Longrightarrow$ check the satisfiability of $\boldsymbol{\varphi} \wedge \neg\boldsymbol{\varphi}[\boldsymbol{\psi} \leftarrow x]$

LTL formula, can be done in PSPACE

**Deciding whether φ is inherently vacuous is in PSPACE**

**lower bound:**

Reduction from **LTL satisfiability**:

$\boldsymbol{\varphi} \equiv \forall x.\boldsymbol{\varphi}[\boldsymbol{\varphi} \leftarrow x]$ iff $\boldsymbol{\varphi} \equiv$ ***false*** .

■

# A Framework for Inherent Vacuity

We create a **general framework** to suit many possible notions of vacuity.

exactly equal/ preserve properties

$$\varphi \equiv \forall x.\varphi[\psi \leftarrow x]$$

over closed/open systems

$\forall$/ $\exists$

all occureces of $\psi$/ a single occurence

The framework is created by using different **parameters**, allowing to work with different definitions of vacuity, different forms of mutations, different contexts (systems), etc.

We begin by extending the notion of inherent vacuity by mutation.

# A Framework for Inherent Vacuity

## The Parameters

**1. Vacuity type:**

Several definitions of vacuity are studied in theory
[Beer et. al. 01], [Kupferman,Vardi 03], [Armoni et. al 03], [Gurfinkel, Chechik 04]

They differ in the mutation:
The semantics of $\forall$

$\forall x.\varphi [\psi \leftarrow x]$

Consider different approaches to occurrences of nonaffecting subformulas.

$\varphi = [\ldots \psi \ldots \psi \ldots \psi]$

etc.

We exemplify: **single occurrence vs. multiple occurrences**

**single occurrences** vs. **multiple occurrences**

**<u>Examples</u>**:

$\boldsymbol{\varphi} = \mathit{grant} \lor (\mathit{up}\ \mathbf{U}\ \mathit{grant})$

The first occurrence of *grant*

$\equiv\ \bot \lor (\mathit{up}\ \mathbf{U}\ \mathit{grant}) \quad \equiv (\mathit{up}\ \mathbf{U}\ \mathit{grant}) \quad \not\equiv\ \forall x.\boldsymbol{\varphi}(\mathit{grant} \leftarrow x)$

The most challenging assignment

or any other subformula of $\boldsymbol{\varphi}$

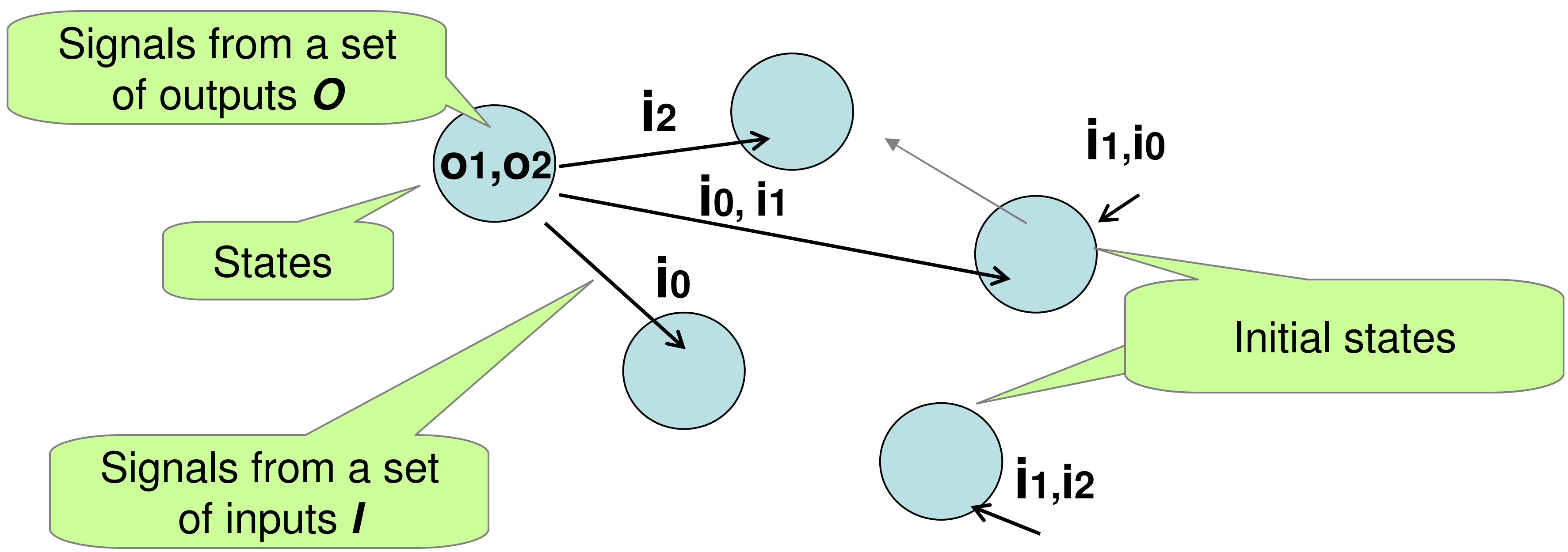
## 2. Equivalence type

The context of ≡ : **closed systems** vs. **open systems**

Kripke structures

Transducers

### Transducer: a reactive system



matches letters and labels read on a path

A computation: $(\mathbf{i}_0, \mathbf{o}_0), (\mathbf{i}_1, \mathbf{o}_1), \ldots$

***T*** realizes $\boldsymbol{\varphi}$

$T \models \boldsymbol{\varphi}$ if all computations of ***T*** satisfy $\boldsymbol{\varphi}$.

Over ***I*** and ***O***

<u>Equivalence of formulas</u>:
**f≡g** if for every model **M**
**M⊨ f iff M⊨g**

equivalence depends on the type of model

**f≡g** in Kripke structures implies **f≡g** in transducers, but not the other way around

[Greimel, Bloem, Jobstmann & Vardi 08]

$\boldsymbol{\equiv_O}$ in transducers is weaker than $\boldsymbol{\equiv_C}$ in Kripke structures

parameter 2

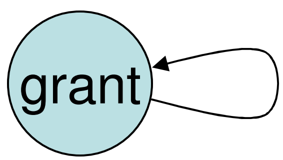
## <u>Example</u>

input : *busy*
output : *grant*

$\boldsymbol{\varphi}=[\mathbf{G}(\mathit{busy} \rightarrow \mathbf{F}(\mathit{grant} \wedge \neg \mathit{busy}))] \vee \mathbf{FG}(\mathit{grant})$

$\boldsymbol{\varphi} \equiv_O \mathit{false}$ (restricts the input)

$\boldsymbol{\varphi} \not\equiv_C \mathit{false}$




$\boldsymbol{\varphi}$ is unrealizable $\Longrightarrow$ $\boldsymbol{\varphi} \equiv_O \forall x.\boldsymbol{\varphi}[\leftarrow x] \equiv_O \mathit{false}$

in the context of **Kripke structures**, $\boldsymbol{\varphi}$ is not inherently vacuous

## 3. Tightening type

**Preserve equivalence** vs. **preserve satisfiability/realizability**

$$\boldsymbol{\varphi} \equiv \forall x . \boldsymbol{\varphi}[\boldsymbol{\psi} \leftarrow x]$$


**<u>Example</u>**

inputs : *busy*
outputs : *ack, grant*

$\boldsymbol{\varphi} = (busy \vee ack) \rightarrow \mathbf{X}\ grant$

$\boldsymbol{\varphi}$ is not inherently vacuous

However, $\forall x . \boldsymbol{\varphi}[ack \leftarrow x] \equiv busy \rightarrow \mathbf{X}\ grant$ is realizable

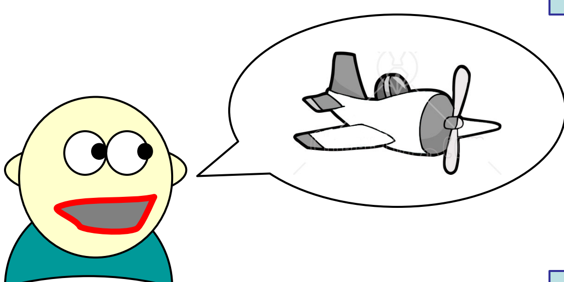
## 4. Polarity type

$\forall x.\varphi[\psi \leftarrow x]$ , a stronger formula vs.

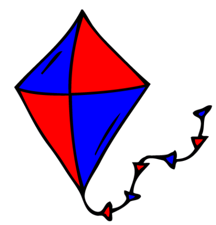
$\exists x.\varphi[\psi \leftarrow x]$ , a weaker formula

In the context of model checking, there is no need to check a weaker formula (it is bound to be satisfied).

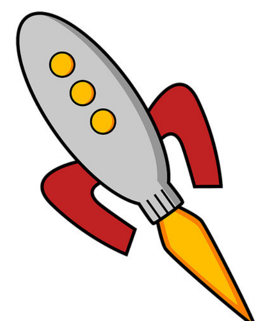
In the context of property-based design, it makes sense!



$\varphi_1 \wedge \varphi_2 \ldots \neg F$ *crash*



$\varphi'_1 \wedge \varphi'_2 \ldots \neg F$ *crash*



## 4. Polarity type

$\forall x.\varphi[\psi \leftarrow x]$ , a stronger formula vs.

$\exists x.\varphi[\psi \leftarrow x]$ , a weaker formula

In the context of model checking, there is no need to check a weaker formula (it is bound to be satisfied)

In the context of property-based design, it makes sense!

**Example**

$\varphi = (\mathbf{F}\ \mathit{grant}) \land (\mathbf{X}\ \mathit{grant}) \equiv \exists x.\varphi\ [\mathbf{F}\ \mathit{grant} \leftarrow x] \equiv \mathbf{X}\ \mathit{grant}$

or : $\equiv \varphi\ [\ \sigma \leftarrow \top]$ for the first occurrence of *grant*

$\varphi$ is not inherently vacuous by strengthening

# Working with the parameters

"$\varphi$ is inherently vacuous of type (V,E,T,P)"

**Vacuity:**

$s_V$ : Single occurrence

$m_V$: Multiple occurences

**Tightening:**

$e_T$ : Equivalent

$p_T$: Preserves satisfaction

**Equivalence:**

$c_E$ : Closed systems

$o_E$: Open systems

**Polarity:**

$s_P$ : Strengthening

$w_P$: Weakening

**Examples:**

$\varphi$ is IV of type ($m_v$, $o_E$ ,$e_T$, $s_p$) if $\varphi \equiv_o \forall x.\ \varphi[\psi \leftarrow x]$

$\varphi$ is IV of type ($s_v$, $c_E$ ,$e_T$, $w_p$) if $\varphi \equiv_c\ \varphi[\sigma \leftarrow \top]$

$\varphi$ is IV of type ($m_v$, $o_E$ ,$p_T$, $s_p$) if $\forall x.\varphi[\psi \leftarrow x]$ is realizable

# Working with the parameters

"$\varphi$ is inherently vacuous of type (V,E,T,P)"

**Vacuity:**

$s_V$ : Single occurrence

$m_V$: Multiple occurences

**Equivalence:**

$c_E$ : Closed systems

$o_E$: Open systems

**Tightening:**

$e_T$ : Equivalent

$p_T$: Preserves satisfaction

**Polarity:**

$s_P$ : Strengthening

$w_P$: Weakening

**Theorem – Connection between types:**

- $(V, E, e_T, s_P) \rightarrow (V, E, p_T, s_P)$ **but** $(V, E, e_T, s_P) \not\leftarrow (V, E, p_T, s_P)$
- $(V, E, e_T, s_P) \neq (V, E, e_T, w_P)$

more can be found in the paper…

## Theorem:

Deciding whether **φ[ψ]** is IV of type (**V**,**E**,**T**,**P**) is PSPACE-Complete for $\mathbf{E}{=}\mathbf{c}_{\mathbf{E}}$

When $\mathbf{V}{=}\mathbf{m}_{\mathbf{V}}$ and $\mathbf{T}{=}\mathbf{p}_{\mathbf{T}}$, it is **EXPSPACE-Complete**

And 2EXPTIME-Complete for $\mathbf{E}{=}\mathbf{o}_{\mathbf{E}}$

## Proof uses:

LTL: satisfiability is **PSPACE**-Complete
realizability is **2EXPTIME**-Complete

∀x.LTL: satisfiability is **EXPSPACE**-Complete

# Refining Inherent Vacuity by Model

Having refined IV **by mutation,** we refine IV

Every structure/some structure

> **φ** is **inherently vacuous by model** if for every Kripke structure **K**,
> if **K**$\models$**φ** then **K** satisfies **φ** vacuously.

over Kripke structures/transducers

all occureces of **ψ**/ a single occurence

**φ** is IV **by model** of type (**V**,**E**,$e_T$,$s_P$) if **φ** is satisfied vacuously in **all** **E**-systems that satisfy **φ**

**φ** is IV **by model** of type (**V**,**E**,$p_T$,$s_P$) if **φ** is satisfied vacuously in **some** **E**-system that satisfies **φ**

The two approaches coincide all over the framwork!

**Theorem**:

**φ** is **inherently vacuous by mutation** of type (**V**,**E**,**T**,**SP**) iff

**φ** is **inherently vacuous by model** of type (**V**,**E**,**T**,**SP**)

# THANK YOU!