UNIVERSITÀ DEGLI STUDI DI BERGAMO

Dipartimento di Ingegneria Gestionale, dell'Informazione e della Produzione

# The Independence Day of Witnessing the Correctness of Systems: From Topological Proofs and Beyond

BCS FACS (Formal Aspects of Computing Science)


Speaker

Claudio MENGHI


Date: 4th July 2023

# Agenda

**2017**

**From Model Checking to a Temporal Proof for Partial Models**
International Conference on Software Engineering and Formal Methods (SEFM)
Bernasconi, Anna; **Menghi, Claudio**; Spoletini, Paola; Zuck, Lenore D; Ghezzi, Carlo

**2020**

**Integrating Topological Proofs with Model Checking to Instrument Iterative Design**
Fundamental Approaches to Software Engineering (FASE)
**Menghi, Claudio**; Rizzi, Alessandro Maria; Bernasconi, Anna

**2021**

**TOrPEDO: Witnessing Model Correctness with Topological Proofs**
Formal Aspects of Computing (FAOC)
**Menghi, Claudio**; Rizzi, Alessandro Maria; Bernasconi, Anna; Spoletini, Paola

**2023**

**Trace Diagnostics for Signal-based Temporal Properties**
IEEE Transactions on Software Engineering (TSE),
Boufaied, Chaima; **Menghi, Claudio**; Bianculli, Domenico; Briand, Lionel C

## 2017

# From Model Checking to a Temporal Proof for Partial Models

International Conference on Software Engineering and Formal Methods (SEFM)

**Bernasconi, Anna**

**Menghi, Claudio**

**Spoletini, Paola**

**Zuck, Lenore D**

**Ghezzi, Carlo**

2017

# From Model Checking to a Temporal Proof for Partial Models

International Conference on Software Engineering and Formal Methods (SEFM)


Bernasconi, Anna

Menghi, Claudio

Spoletini, Paola

Zuck, Lenore D

Ghezzi, Carlo


# Introduction

# Genesis

**Model Checking** and **Theorem Proving** are two techniques proposed to **help** designers and developers in producing a software that is correct

*From model checking to a temporal proof.*
*Peled, Doron, and Lenore Zuck.*
*Proceedings of the 8th international SPIN workshop on Model checking of software. 2001.*

# Genesis

### *Model Checking*

M: model of the system

$\phi$: property of interest

$$M \models \phi$$

yes

no + counterexample

### *Theorem Proving*

M: model of the system

$\phi$: property of interest

$$M \models \phi$$

yes + proof

no

*From model checking to a temporal proof.*
*Peled, Doron, and Lenore Zuck.*
*Proceedings of the 8th international SPIN workshop on Model checking of software. 2001.*

# Preliminaries

***Model Checking + Theorem Proving***

M: model of the system

$\phi$: property of interest

$$M \models \phi$$

yes + proof

no + counterexample

*From model checking to a temporal proof.*
*Peled, Doron, and Lenore Zuck.*
*Proceedings of the 8th international SPIN workshop on Model checking of software. 2001.*

## Preliminaries

***Model Checking + Theorem Proving***

M: model of the system

$\phi$: property of interest

***Assumption:*** the model M of the system is completely specified, i.e., it is a **definitive model**

*From model checking to a temporal proof.*
*Peled, Doron, and Lenore Zuck.*
*Proceedings of the 8th international SPIN workshop on Model checking of software. 2001.*

# Partial Models

However, in practice, models can be only
partially specified or incomplete

# Partial Models (*Formal Methods)*

- A modal process logic
  Larsen, Kim G., and Bent Thomsen.
  Logic in Computer Science, 1988
- Model checking partial state spaces with 3-valued temporal logics
  G Bruns, P Godefroid
  Computer Aided Verification,1999
- Multi-valued model checking via classical model checking.
  Gurfinkel, Arie, and Marsha Chechk.
  Lecture notes in computer science 2003
- Dealing with Incompleteness in Automata-Based Model Checking
  C Menghi, P Spoletini, C Ghezzi
  Formal Methods, 2016

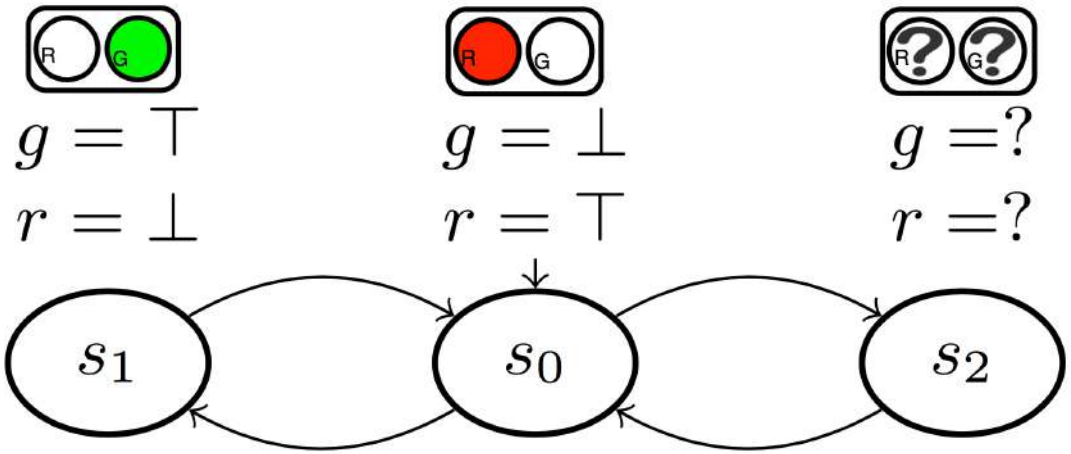
# Partial Models (*Software Engineering)*

- Managing design-time uncertainty
  Michalis Famelis· Marsha Chechik.
  Software & Systems Modeling, 2017.
- Partial models: Towards modeling and reasoning with uncertainty
  M Famelis, R Salay, M Chechik
  Software Engineering (ICSE), 2012
- Synthesis of partial behavior models from properties and scenarios
  S Uchitel, G Brunet, M Chechik
  IEEE Transactions on Software Engineering, 2009

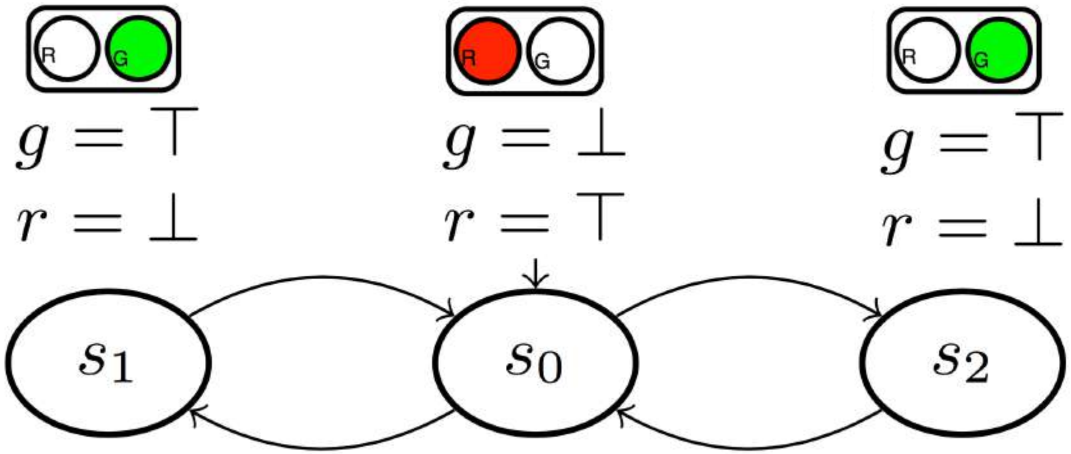
# Partial Models (*Requirements Engineering)*

- Supporting early decisionmaking in the presence of uncertainty.
  Horkoff, J., Salay, R., Chechik, M., Di Sandro, A.:
  Requirements Engineering Conference, 2014
- Integrating Goal Model Analysis with Iterative Design
  C Menghi, P Spoletini, C Ghezzi
  International Working Conference on Requirements Engineering:
  Foundation for Software Quality, 2017

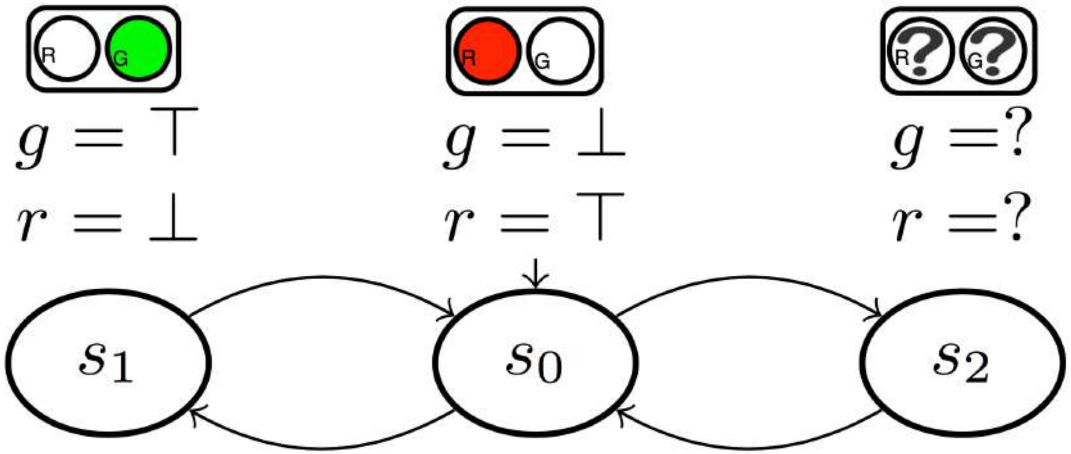
# Running Example



# Running Example



# Running Example




- Red lights up infinitely often

$$\phi_1 = \Box\Diamond red.$$

- Green lights up infinitely often

$$\phi_2 = \Box\Diamond green.$$

- When the light is red, it will always be green

$$\phi_3 = \Box(red \rightarrow \Box green)$$

# Problem Statement

**Question**

How to ***help*** designers in producing *correct* software with model checking and theorem providing results for partial models?

*From model checking to a temporal proof.*
*Peled, Doron, and Lenore Zuck.*
*Proceedings of the 8th international SPIN workshop on Model checking of software. 2001.*

2017

# From Model Checking to a Temporal Proof for Partial Models

International Conference on Software Engineering and Formal Methods (SEFM)


Bernasconi, Anna

Menghi, Claudio

Spoletini, Paola

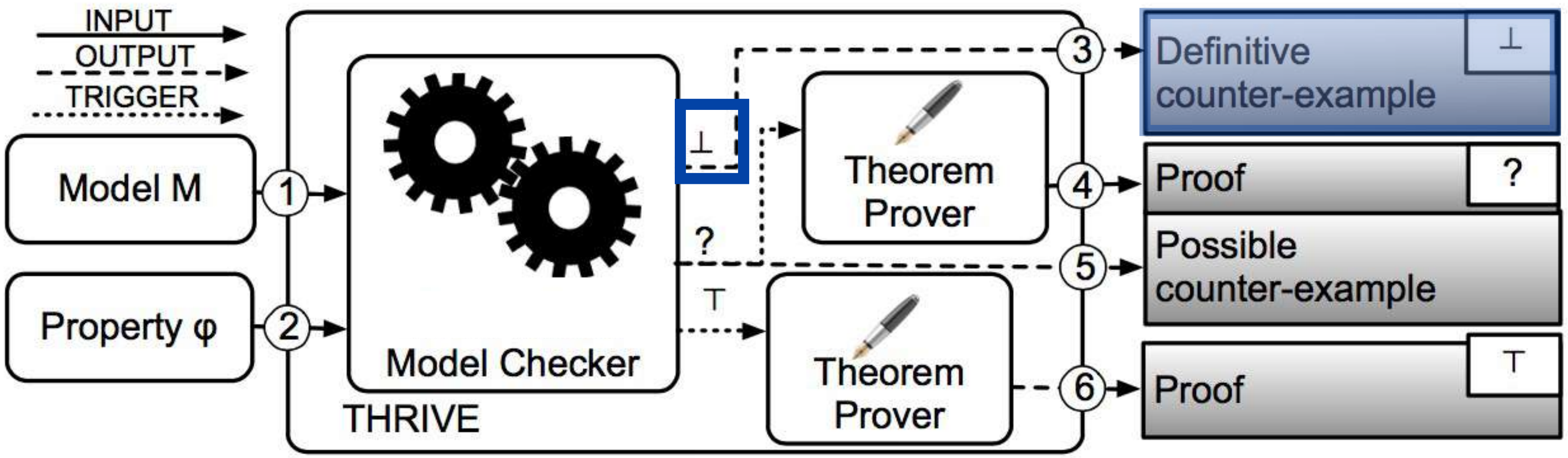
Zuck, Lenore D

Ghezzi, Carlo


## Contribution

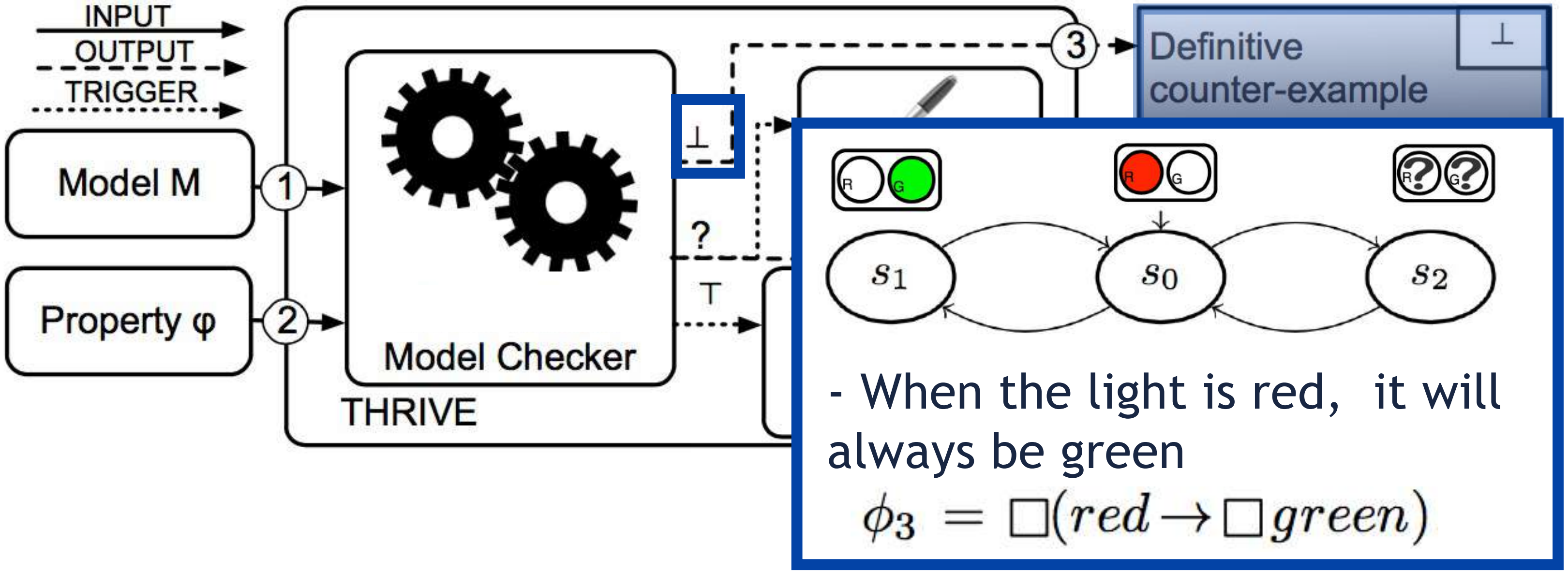
# Contribution (THRIVE)

- THRIVE: THRee valued Integrated Verification framEwork for partial models.



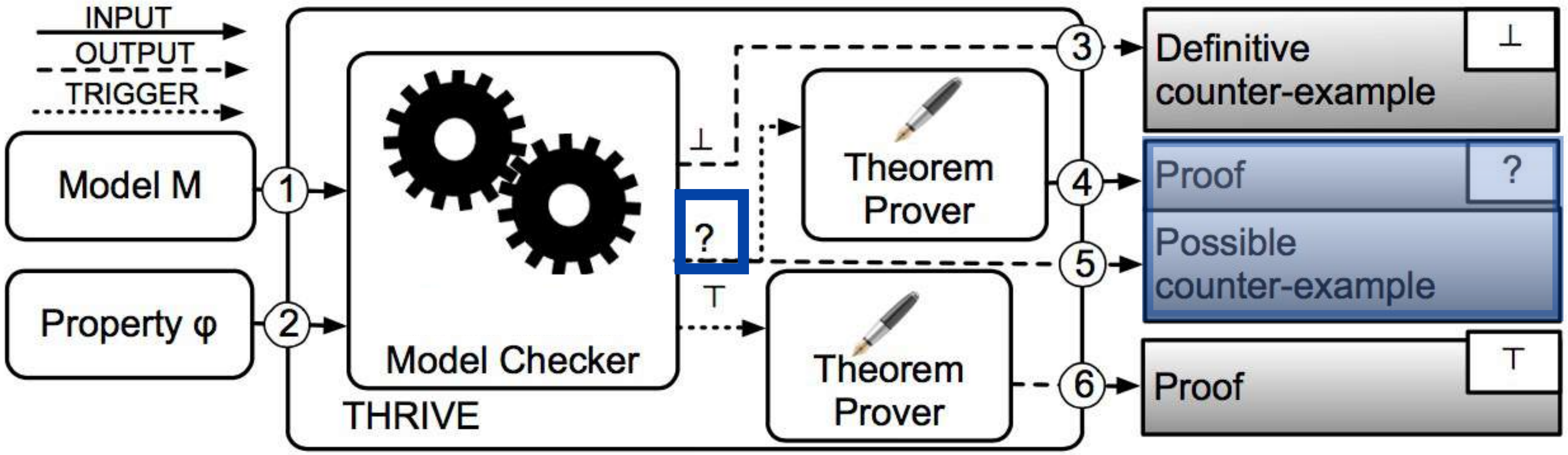
# Contribution (THRIVE)

- THRIVE: THRee valued Integrated Verification framEwork for partial models.



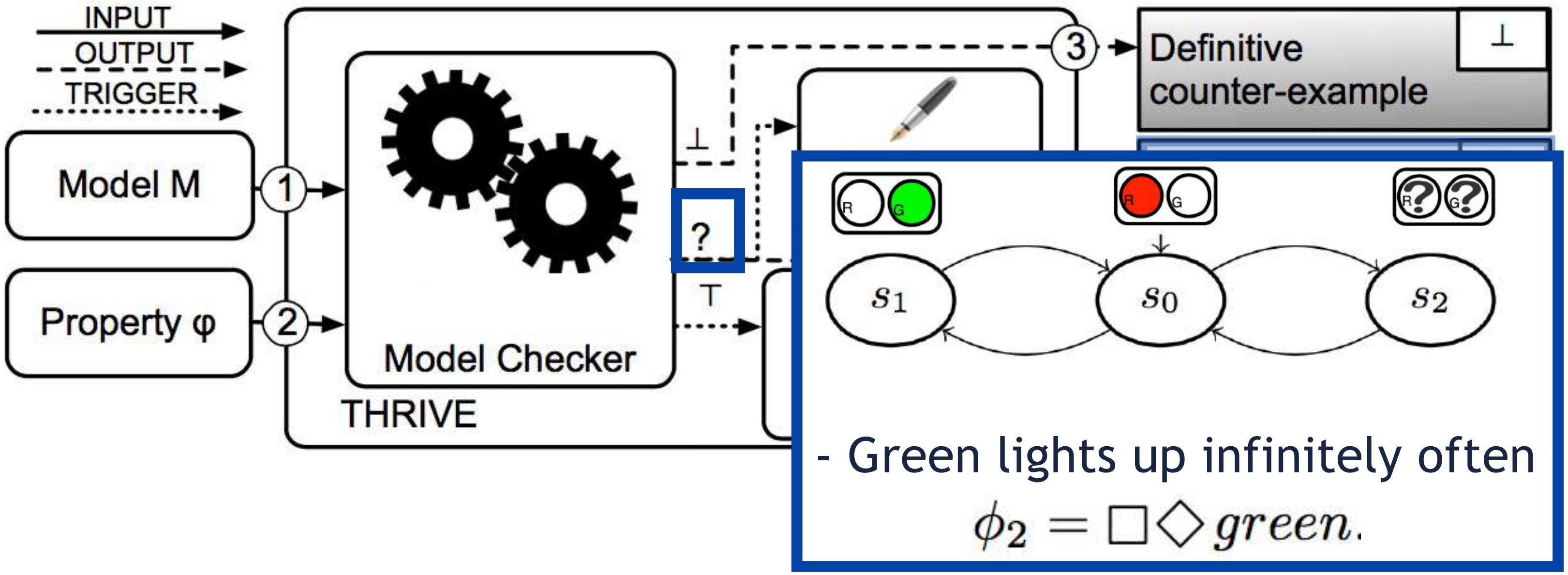
# Contribution (THRIVE)

- THRIVE: THRee valued Integrated Verification framEwork for partial models.



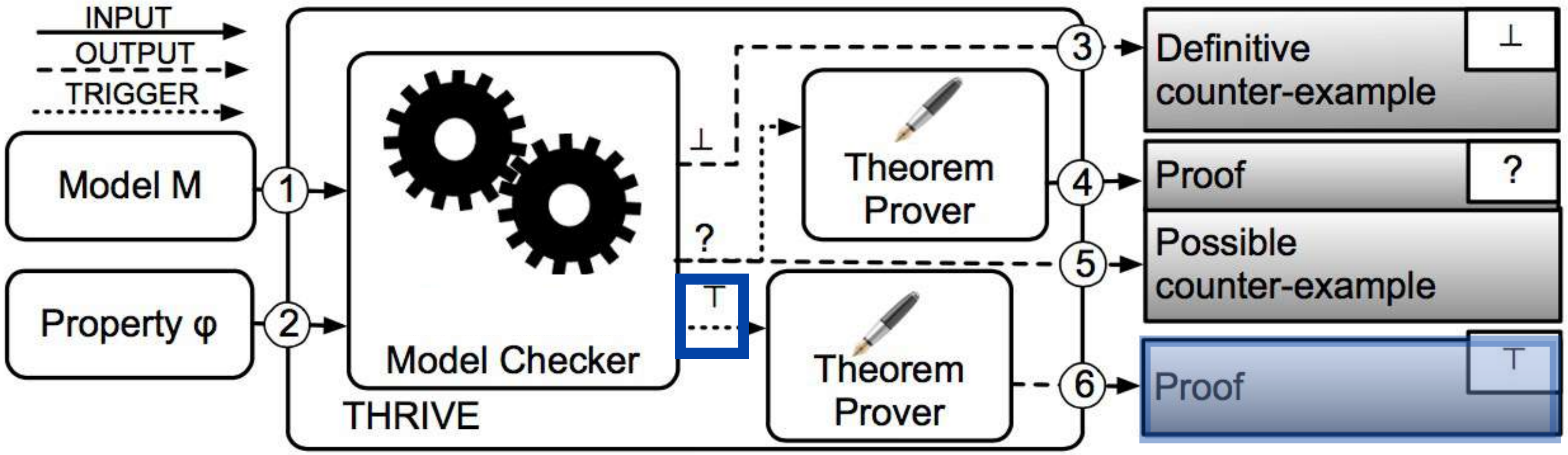
# Contribution (THRIVE)

- THRIVE: THRee valued Integrated Verification framEwork for partial models.



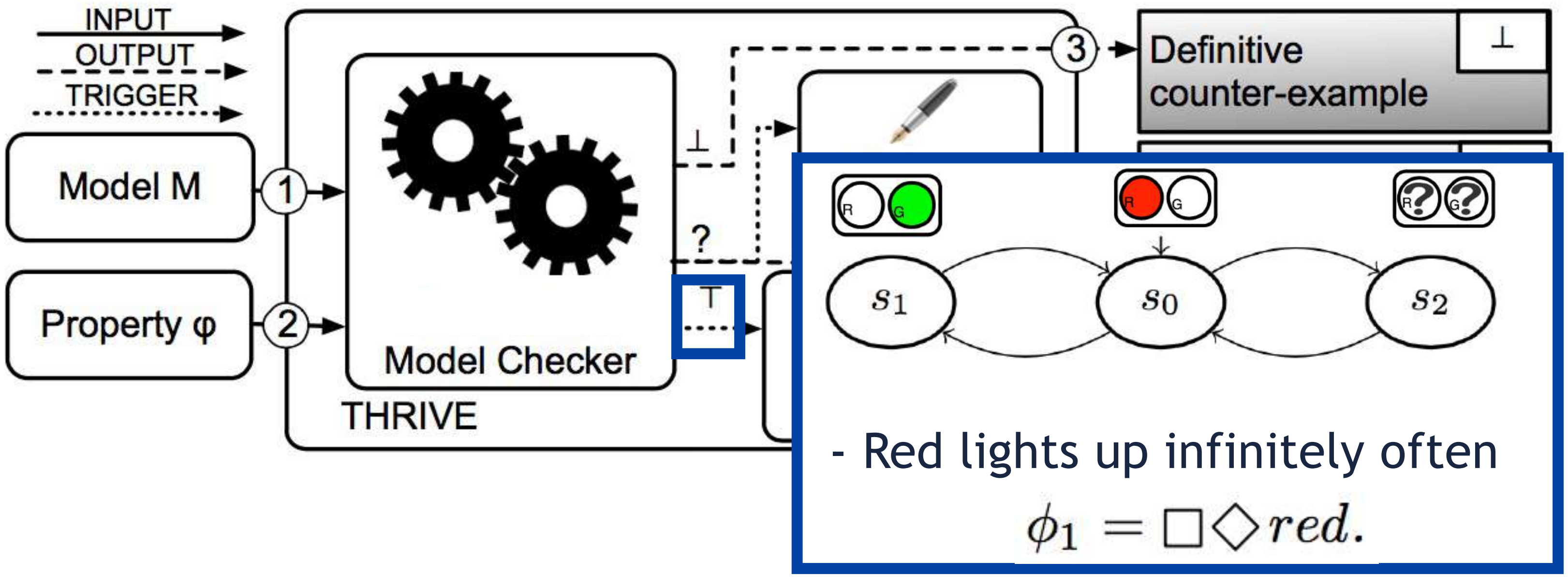
# Contribution (THRIVE)

- THRIVE: THRee valued Integrated Verification framEwork for partial models.



# Contribution (THRIVE)

- THRIVE: THRee valued Integrated Verification framEwork for partial models.



2017

## From Model Checking to a Temporal Proof for Partial Models

International Conference on Software Engineering and Formal Methods (SEFM)


Bernasconi, Anna

Menghi, Claudio

Spoletini, Paola

Zuck, Lenore D

Ghezzi, Carlo


# An Instance of THRIVE

# An instance of THRIVE

- Model of the system:
  Partial Kripke Structures (PKS)

- Property of interest:
  Linear Time Temporal Logic (LTL)

# An instance of THRIVE

- Two possible semantics of LTL over PKS can be considered
  - *Three-valued semantics*: it is based on information ordering T>?>⊥
  - *Thorough semantics*: it is based on the notion of refinement

*Model checking partial state spaces with 3-valued temporal logics.*
Bruns, G., Godefroid, P.
CAV 1999

*Generalized model checking: reasoning about partial state spaces*
Bruns, G., Godefroid, P.
CONCUR 2000

# An instance of THRIVE: Model checking

Two possible semantics of LTL over PKS can be considered

| | Model checking | Result |
|---|---|---|
| Three-Valued | faster (it exploits two runs of classical model checkers) | Not "correct" when ? is returned |
| Thorough | slower (it requires more complex verification procedures) | Correct |

*Model checking partial state spaces with 3-valued temporal logics.*
Bruns, G., Godefroid, P.
CAV 1999

*Generalized model checking: reasoning about partial state spaces*
Bruns, G., Godefroid, P.
CONCUR 2000

# An instance of THRIVE: Model checking

Two possible semantics of LTL over PKS can be considered

| | Model checking | Result |
|---|---|---|
| **Three-Valued** | faster (it exploits two runs of classical model checkers) | Not "correct" when ? is returned |
| **Thorough** | slower (it requires more complex verification procedures) | Correct |

*Model checking partial state spaces with 3-valued temporal logics.*
Bruns, G., Godefroid, P.
CAV 1999

*Generalized model checking: reasoning about partial state spaces*
Bruns, G., Godefroid, P.
CONCUR 2000

# An instance of THRIVE: Model checking

Two possible semantics of LTL over PKS can be considered

| | Model checking | Result |
|---|---|---|
| **Three-Valued** | faster (it exploits two runs of classical model checkers) | Not "correct" when ? is returned |
| **Thorough** | slower (it requires more complex verification procedures) | Correct |

*Model checking partial state spaces with 3-valued temporal logics.*
Bruns, G., Godefroid, P.
CAV 1999

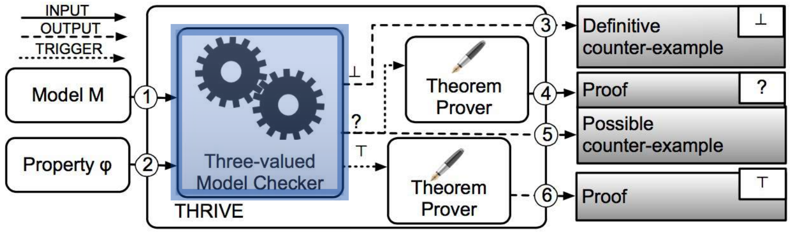
*Generalized model checking: reasoning about partial state spaces*
Bruns, G., Godefroid, P.
CONCUR 2000

# An instance of THRIVE: Model checking

The three-valued model checking can be solved as follows

$$[(M, s) \models \phi] = \begin{cases} \top & \text{if } (M_{pes}, s) \models \phi \\ \bot & \text{if } (M_{opt}, s) \not\models \phi \\ ? & otherwise \end{cases}$$

*Model checking partial state spaces with 3-valued temporal logics.*
Bruns, G., Godefroid, P.
CAV 1999



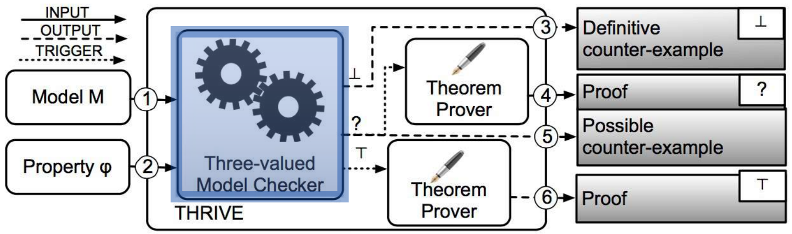
# An instance of THRIVE: Model checking

The three-valued model checking can be solved as follows

$$[(M, s) \models \phi] = \begin{cases} \top & \text{if } (M_{pes}, s) \models \phi \\ \bot & \text{if } (M_{opt}, s) \not\models \phi \\ ? & otherwise \end{cases}$$

I do my best to violate the property

*Model checking partial state spaces with 3-valued temporal logics.*
Bruns, G., Godefroid, P.
CAV 1999



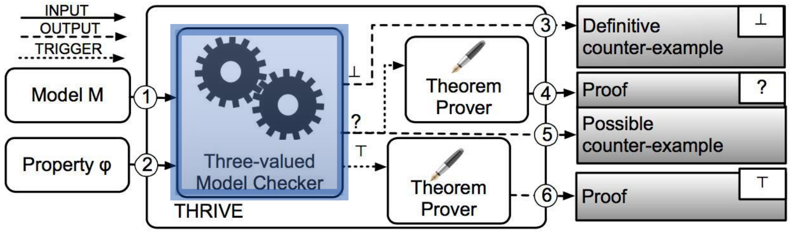
# An instance of THRIVE: Model checking

The three-valued model checking can be solved as follows

$$[(M, s) \models \phi] = \begin{cases} \top & \text{if } (M_{pes}, s) \models \phi \\ \bot & \text{if } (M_{opt}, s) \not\models \phi \\ ? & otherwise \end{cases}$$

I do my best to satisfy the property

*Model checking partial state spaces with 3-valued temporal logics.*
Bruns, G., Godefroid, P.
CAV 1999



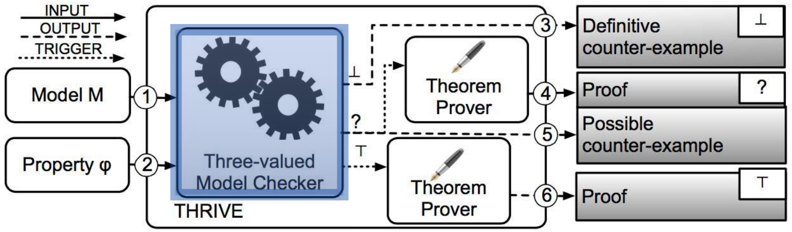
# An instance of THRIVE: Model checking

The three-valued model checking can be solved as follows

$$[(M, s) \models \phi] = \begin{cases} \top & \text{if } (M_{pes}, s) \models \phi \\ \bot & \text{if } (M_{opt}, s) \not\models \phi \\ ? & \textit{otherwise} \end{cases}$$

If none of the previous condition holds

*Model checking partial state spaces with 3-valued temporal logics.*
Bruns, G., Godefroid, P.
CAV 1999

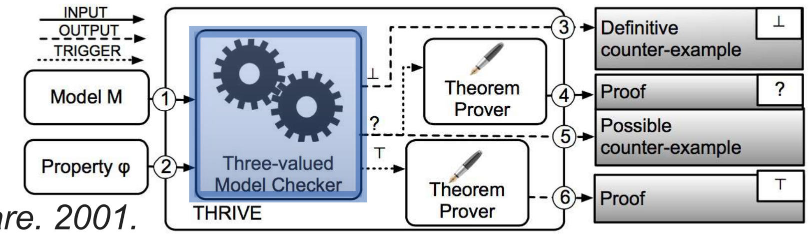


# An instance of THRIVE: Theorem Proving

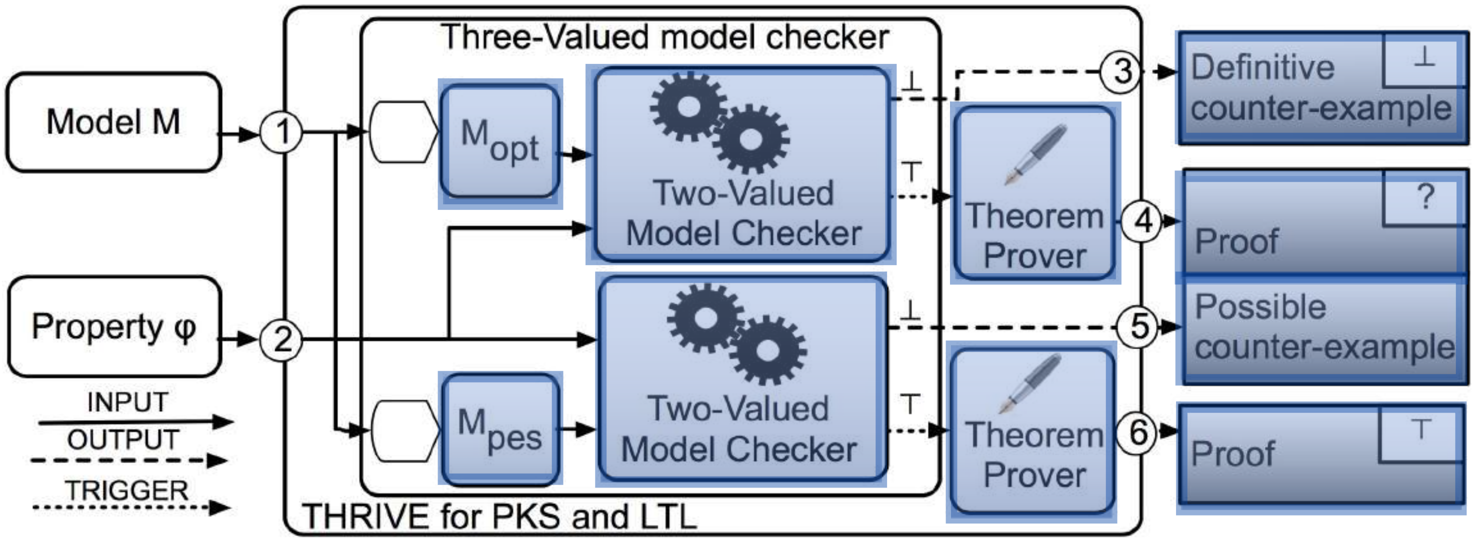
The ***deductive verification*** framework produces a **proof** which explains why M ⊨ φ

- it identifies ***failed states***
- it applies a set of **deduction rules** (*successors, induction, conjunction rule)*

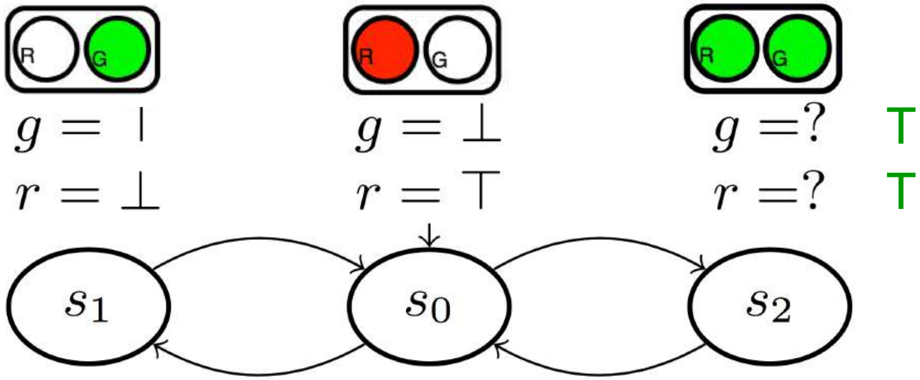
*From model checking to a temporal proof.*
*Peled, Doron, and Lenore Zuck.*
*Proceedings of the 8th international SPIN workshop on Model checking of software. 2001.*

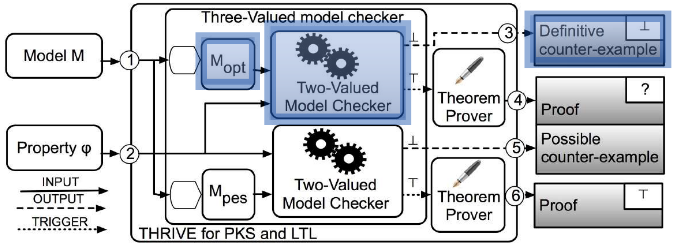


# An instance of THRIVE



# An instance of THRIVE: Running example



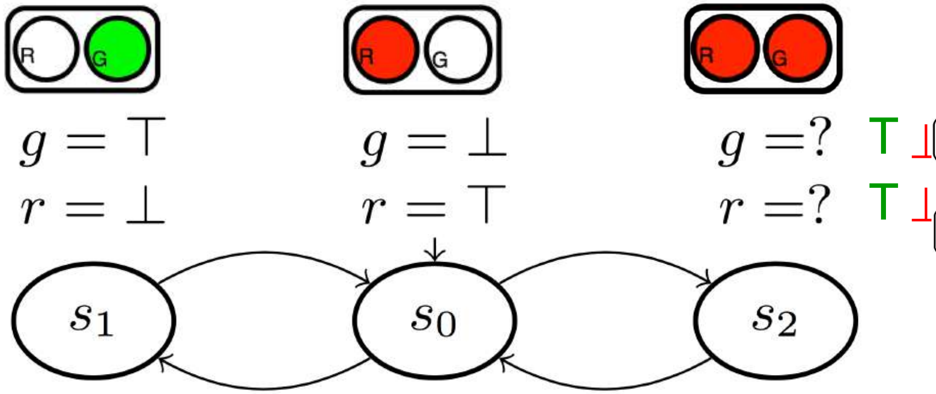



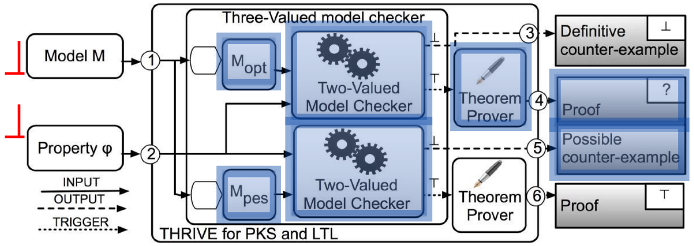

- When the light is red, it will always be green

$$\phi_3 = \Box(red \rightarrow \Box green)$$

$$counterexample \ (s_0, s_1)^\omega$$

# An instance of THRIVE: Running example




- Green lights up infinitely often

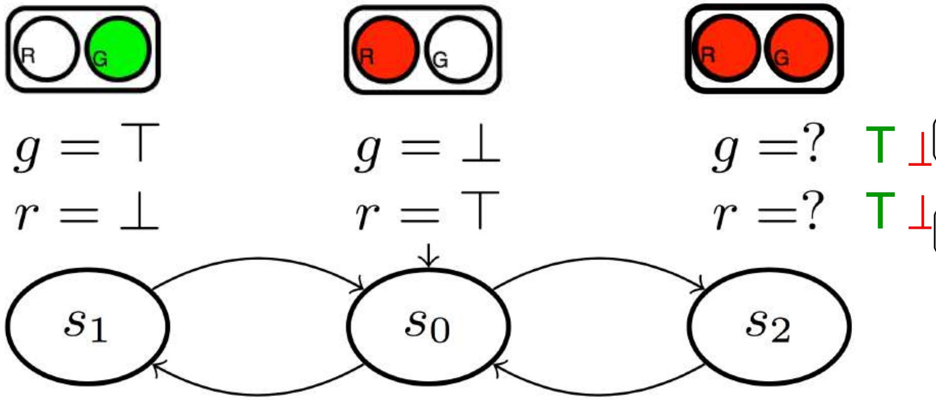
$$\phi_2 = \Box \Diamond green.$$

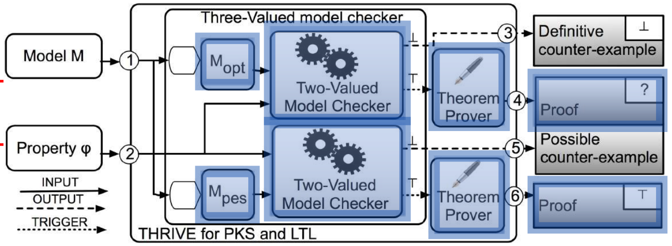
| *Step 1* | *Step 2* | *Step 3* | *Step 4* |
|---|---|---|---|
| **Fail** | **Successors** | **Induction** | **Conjunction** |
| $\langle s_1, q_1\rangle \in \mathcal{F}(I_{opt})$<br>$\langle s_2, q_1\rangle \in \mathcal{F}(I_{opt})$<br>———<br>$s_1, s_2 \models g \vee \bigcirc \Diamond g$ | $s_0 \rightarrow \{s_1, s_2\}$<br>$s_1 \models g \vee \bigcirc \Diamond g$<br>$s_2 \models g \vee \bigcirc \Diamond g$<br>———<br>$s_0 \models g \vee \bigcirc \Diamond g$ | $s_0, s_1, s_2 \models g \vee \bigcirc \Diamond g$<br>$s_0 \rightarrow \{s_1, s_2\}$<br>$s_1 \rightarrow \{s_0\}$<br>$s_2 \rightarrow \{s_0\}$<br>———<br>$s_0, s_1, s_2 \models \bigcirc \Box \Diamond g$ | $s_0 \models \bigcirc \Box \Diamond g$<br>$s_0 \models g \vee \bigcirc \Diamond g$<br>$\bigcirc \Box \Diamond g \wedge (g \vee \bigcirc \Diamond g) \rightarrow \phi_2$<br>———<br>$s_0 \models \phi_2$ |

$$(s_0, s_2)^\omega$$

*possible counterexample*

# An instance of THRIVE: Running example




- Red lights up infinitely often

$$\phi_1 = \Box \Diamond red.$$

| *Step 1* Fail | *Step 2* Successors | *Step 3* Induction | *Step 4* Conjunction |
|---|---|---|---|
| $\langle s_1, q_1\rangle \in \mathcal{F}(I_{opt})$<br>$\langle s_2, q_1\rangle \in \mathcal{F}(I_{opt})$ | $s_0 \rightarrow \{s_1, s_2\}$<br>$s_1 \models r \vee \bigcirc \Diamond r$<br>$s_2 \models r \vee \bigcirc \Diamond r$ | $s_0, s_1, s_2 \models r \vee \bigcirc \Diamond r$<br>$s_0 \rightarrow \{s_1, s_2\}$<br>$s_1 \rightarrow \{s_0\}$<br>$s_2 \rightarrow \{s_0\}$ | $s_0 \models \bigcirc \Box \Diamond r$<br>$s_0 \models r \vee \bigcirc \Diamond r$<br>$\bigcirc \Box \Diamond r \wedge (r \vee \bigcirc \Diamond r) \rightarrow \phi_2$ |
| $s_1, s_2 \models r \vee \bigcirc \Diamond r$ | $s_0 \models r \vee \bigcirc \Diamond r$ | $s_0, s_1, s_2 \models \bigcirc \Box \Diamond r$ | $s_0 \models \phi_2$ |

# An instance of THRIVE: Model checking

Two possible semantics of LTL over PKS can be considered

| | Model checking | Result |
|---|---|---|
| **Three-Valued** | faster (it exploits two runs of classical model checkers) | Not correct when ? is returned |
| **Thorough** | slower (it requires more complex verification procedures) | Correct |

*Model checking partial state spaces with 3-valued temporal logics.*
Bruns, G., Godefroid, P.
CAV 1999

*Generalized model checking: reasoning about partial state spaces*
Bruns, G., Godefroid, P.
CONCUR 2000

# An instance of THRIVE: Correctness

- What about the **thorough** semantics?
  - In many practically interesting cases, the thorough semantics is **not more precise** than the three-valued*
  - If the LTL formula is ***Self-minimizing*** the result is correct**

* How thorough is thorough enough?
Gurfinkel, A., Chechik, M.
CHARME 2005

**Model checking vs. generalized model checking: semantic minimizations for temporal logics
Godefroid, P., Huth, M.
Logic in Computer Science, 2005

# An instance of THRIVE:  Correctness

- most of the **patterns** proposed in literature are expressed using self-minimising formulae *
- if satisfies some constraints **(sufficient conditions)** then it is self-minimizing **

* Model checking vs. generalized model checking: semantic minimizations for temporal logics.
  Godefroid, P., Huth, M.
  Logic in Computer Science

** Efficient patterns for model checking partial state spaces in CTL ∩ LTL
  Antonik, A., Huth, M
  Notes Theor. Comput. Sci

2017

## From Model Checking to a Temporal Proof for Partial Models

International Conference on Software Engineering and Formal Methods (SEFM)

Bernasconi, Anna

Menghi, Claudio

Spoletini, Paola

Zuck, Lenore D

Ghezzi, Carlo

# Preliminary Evaluation

# Preliminary Evaluation

**RQ**: How **effective** is THRIVE w.r.t. incremental development?

# Preliminary Evaluation

- we simulated the design of a critical software system*
- the system is used by physicians to check visual problems

* P. Arcaini, S. Bonfanti, A. Gargantini, A. Mashkoor, and E. Riccobene.
Formal validation and verification of a medical software critical component.
In Formal Methods and Models for Codesign, pages 80–89. IEEE, 2015.

# Preliminary Evaluation

- We designed three properties that the system has to satisfy following well-known property patterns**
- We created an abstraction of the final model
- We checked how THRIVE supports incremental development

** M. B. Dwyer, G. S. Avrunin, and J. C. Corbett.
Property specification patterns for finite-state verification.
In Proceedings of the second workshop on Formal methods in software practice, pages 7–15. ACM, 1998.

# Preliminary Evaluation

For property ψ1, THRIVE returns a definitive counterexample showing the reason for the violation.

**The property is wrong.**

# Preliminary Evaluation

For property ψ2, THRIVE returns the T value, since the property is satisfied.

The proof enabled us understanding the reason for the satisfaction.

# Preliminary Evaluation

For property ψ3, THRIVE returns the value ? and

- a possible counterexample shows the violation for the pessimistic approximation
- The possible proof shows why the property of interest is satisfied on the optimistic approximation

2017

# From Model Checking to a Temporal Proof for Partial Models

International Conference on Software Engineering and Formal Methods (SEFM)


Bernasconi, Anna

Menghi, Claudio

Spoletini, Paola

Zuck, Lenore D

Ghezzi, Carlo


# Lessons learned

# Lessons learned

Creating new instances of THRIVE is **not easy**!

- Choose/define a ***semantics*** of formulae on partial models is not easy
- it influences the model checker and the theorem improving that can be used

## Lessons learned

- The selection of the **model checkers** and the **theorem proving** to be combined must be done carefully to ensure the **correctness** of the obtained framework
- The selected model checker/theorem prover may be **changed** to be successfully combined

2017

## From Model Checking to a Temporal Proof for Partial Models

International Conference on Software Engineering and Formal Methods (SEFM)

Bernasconi, Anna

Menghi, Claudio

Spoletini, Paola

Zuck, Lenore D

Ghezzi, Carlo

# Conclusions

# Conclusions and Future Work

- We propose THRIVE
- We show an instance of THRIVE that considers PKS and LTL
- We assess effectiveness on a simulated experiment

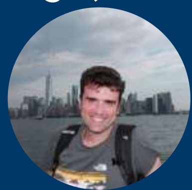
# Conclusions and Future Work

**Future Work**: integrate THRIVE on top of existing theorem provers and model checkers

# Integrating Topological Proofs with Model Checking to Instrument Iterative Design

Fundamental Approaches to Software Engineering (FASE)


Menghi, Claudio



Rizzi, Alessandro Maria

Bernasconi, Anna


2020


UNIVERSITÀ DEGLI STUDI DI BERGAMO

Dipartimento di Ingegneria Gestionale, dell'Informazione e della Produzione

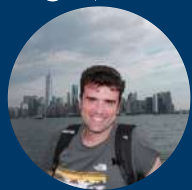
# Introduction

## Integrating Topological Proofs with Model Checking to Instrument Iterative Design

Fundamental Approaches to Software Engineering (FASE)

2020

Menghi, Claudio



Rizzi, Alessandro Maria

Bernasconi, Anna

# Motivation

**THRIVE: THRee valued Integrated Verification framEwork for partial models.**

## Model Checking + Theorem Proving

M: **partial** model

φ: property

$M \models \varphi$

No (⊥) + counterexample

Yes (⊤) + definitive proof

Maybe (?) + possible counterexample and proof

From model checking to a temporal proof for partial models
A Bernasconi, C Menghi, P Spoletini, LD Zuck, C Ghezzi
International Conference on Software Engineering and Formal Methods (SEFM), 2017

# Motivation

**THRIVE: THRee valued Integrated Verification framEwork for partial models.**

## Model Checking + Theorem Proving

M: **partial** model

φ: property

$M \models \varphi$

No (⊥) + counterexample

Yes (⊤) + definitive proof

Maybe (?) + possible counterexample and proof

**Deductive Proofs**

From model checking to a temporal proof for partial models
A Bernasconi, C Menghi, P Spoletini, LD Zuck, C Ghezzi
International Conference on Software Engineering and Formal Methods (SEFM), 2017

# Motivation

Deductive proofs

- are usually difficult to understand
- their size significantly grows with the size of the model analysed

# Motivation

How could we provide more effective support and guidance to engineers when properties of interest are satisfied or possibly satisfied?

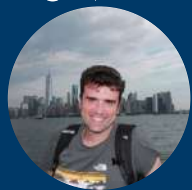
# Running Example

## Integrating Topological Proofs with Model Checking to Instrument Iterative Design

Fundamental Approaches to Software Engineering (FASE)

2020

Menghi, Claudio



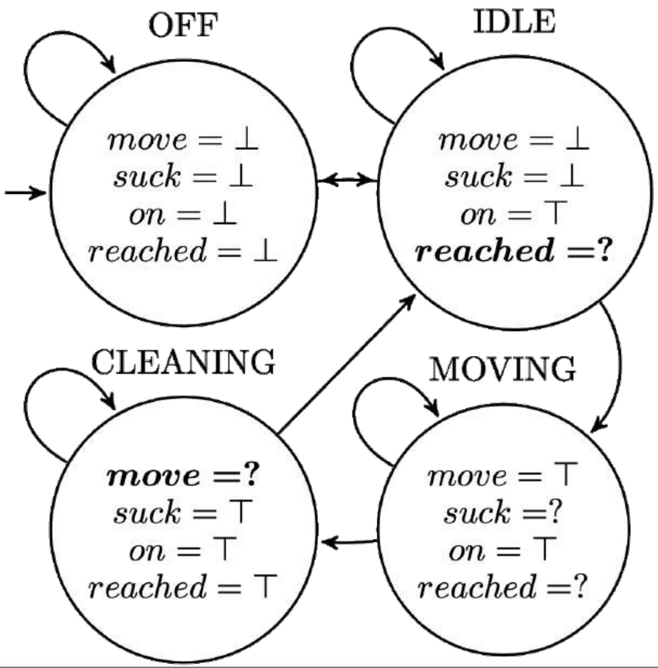
Rizzi, Alessandro Maria

Bernasconi, Anna

# Vacuum-cleaner robot

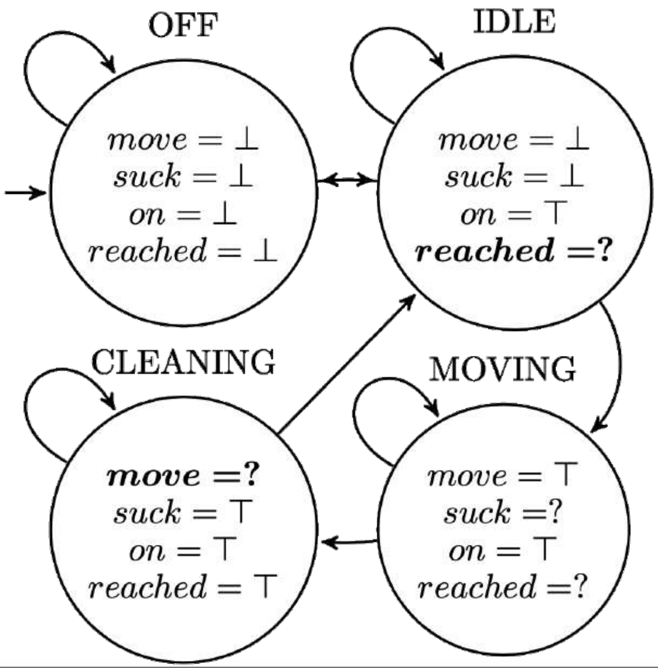
| Textual Requirements | LTL formulae |
|---|---|
| $\phi_1$: the robot is drawing dust (*suck*) only if it has *reached* the cleaning site. | $\phi_1 \equiv \mathcal{G}(suck \rightarrow reached)$ |
| $\phi_2$: the robot must be turned *on* before it can *move*. | $\phi_2 \equiv \mathcal{G}((\neg move)\, \mathcal{W}\, on)$ |
| $\phi_3$: if the robot is *on* and stationary ($\neg move$), it must be drawing dust (*suck*). | $\phi_3 \equiv \mathcal{G}(((\neg move) \land on) \rightarrow suck)$ |
| $\phi_4$: the robot must *move* before it is allowed to draw dust (*suck*). | $\phi_4 \equiv ((\neg suck)\, \mathcal{W}(move \land (\neg suck)))$ |

# Vacuum-cleaner robot: Initial Design




| **Textual Requirements** | **LTL formulae** |
|---|---|
| $\phi_1$: the robot is drawing dust (*suck*) only if it has *reached* the cleaning site. | $\phi_1 \equiv \mathcal{G}(suck \rightarrow reached)$ |
| $\phi_2$: the robot must be turned *on* before it can *move*. | $\phi_2 \equiv \mathcal{G}((\neg move)\,\mathcal{W}\,on)$ |
| $\phi_3$: if the robot is *on* and stationary ($\neg move$), it must be drawing dust (*suck*). | $\phi_3 \equiv \mathcal{G}(((\neg move) \wedge on) \rightarrow suck)$ |
| $\phi_4$: the robot must *move* before it is allowed to draw dust (*suck*). | $\phi_4 \equiv ((\neg suck)\,\mathcal{W}(move \wedge (\neg suck)))$ |

# Vacuum-cleaner robot: Initial Design




⊥: violated

⊤: satisfied

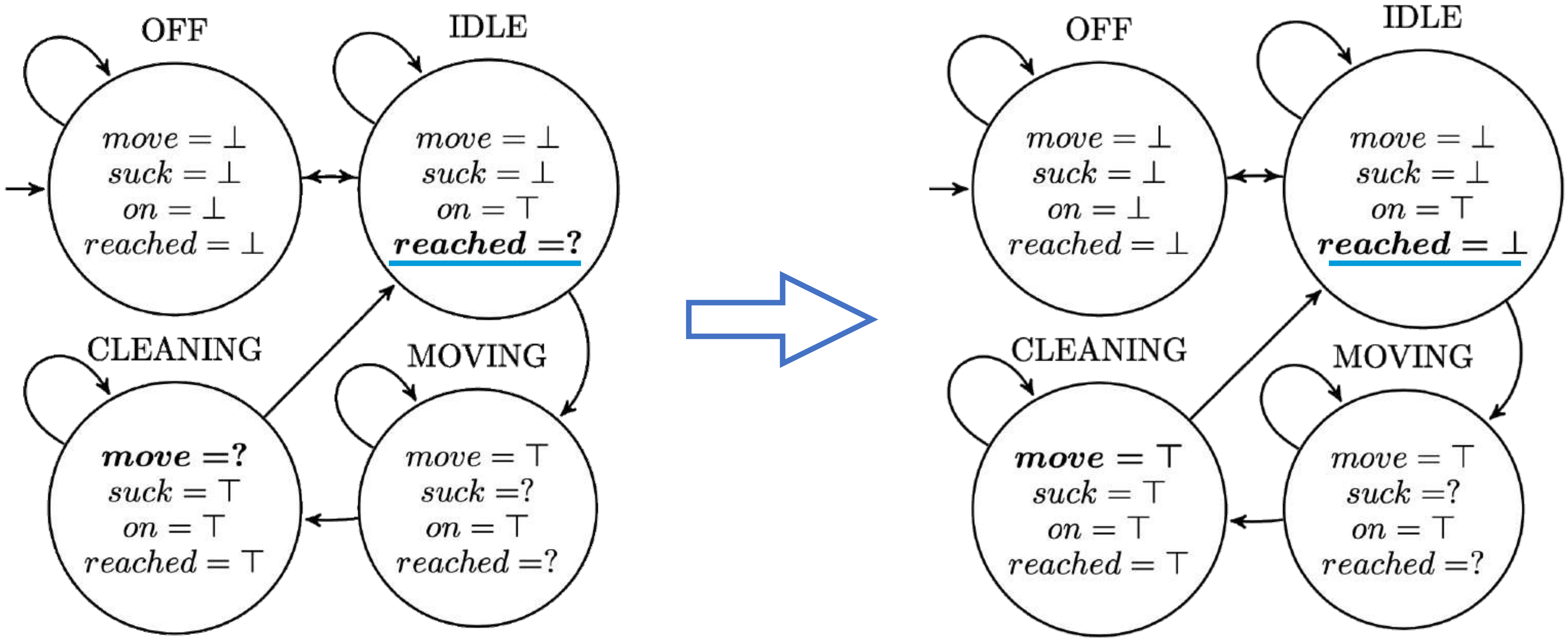
?: possibly satisfied

| Textual Requirements | LTL formulae | |
|---|---|---|
| $\phi_1$: the robot is drawing dust (*suck*) only if it has *reached* the cleaning site. | $\phi_1 \equiv \mathcal{G}(suck \rightarrow reached)$ | ? |
| $\phi_2$: the robot must be turned *on* before it can *move*. | $\phi_2 \equiv \mathcal{G}((\neg move)\, \mathcal{W}\, on)$ | ⊤ |
| $\phi_3$: if the robot is *on* and stationary ($\neg move$), it must be drawing dust (*suck*). | $\phi_3 \equiv \mathcal{G}(((\neg move) \wedge on) \rightarrow suck)$ | ⊥ |
| $\phi_4$: the robot must *move* before it is allowed to draw dust (*suck*). | $\phi_4 \equiv ((\neg suck)\, \mathcal{W}(move \wedge (\neg suck)))$ | ? |

# Vacuum-cleaner robot: Revision

- During a revision, an engineer can:
  - add/remove states
  - add/remove transitions
  - change the values of the propositions

# Vacuum-cleaner robot: Revision



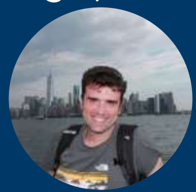
# TOrPEDO: Overview

## Integrating Topological Proofs with Model Checking to Instrument Iterative Design

Fundamental Approaches to Software Engineering (FASE)

2020


Menghi, Claudio



Rizzi, Alessandro Maria

Bernasconi, Anna

# Topological Proofs

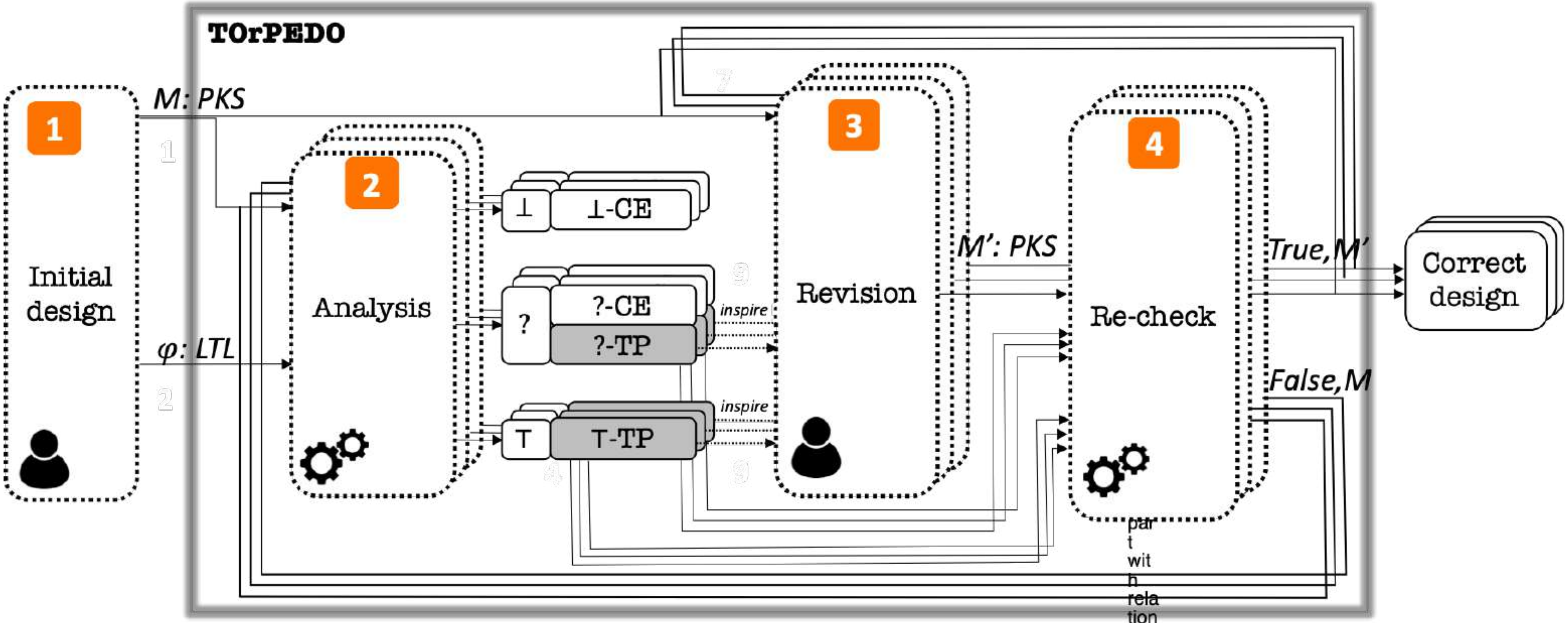
A topological proof is a slice of the model
that witnesses property satisfaction

# Topological Proofs

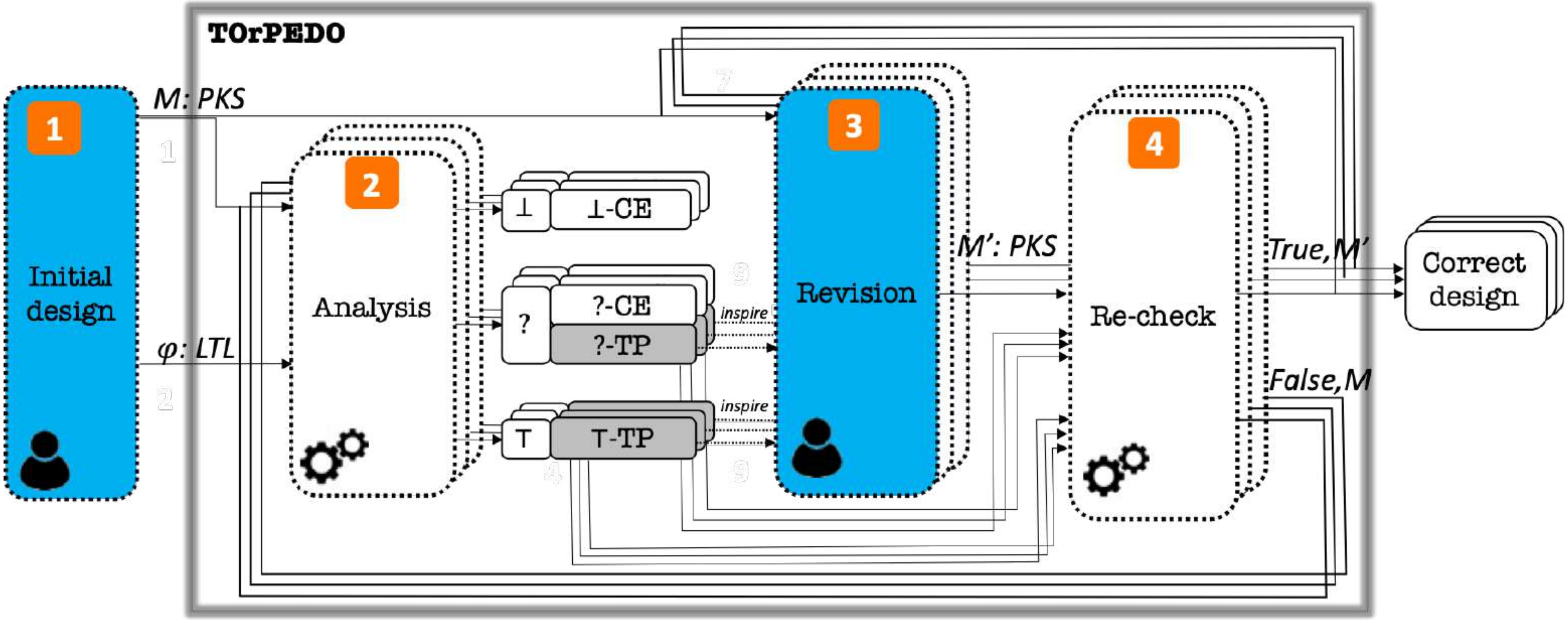
A topological proof is a slice of the model that witnesses property satisfaction

If the engineer does not modify elements of the models in the topological proof, then the revision will not violate the property

# TOrPEDO

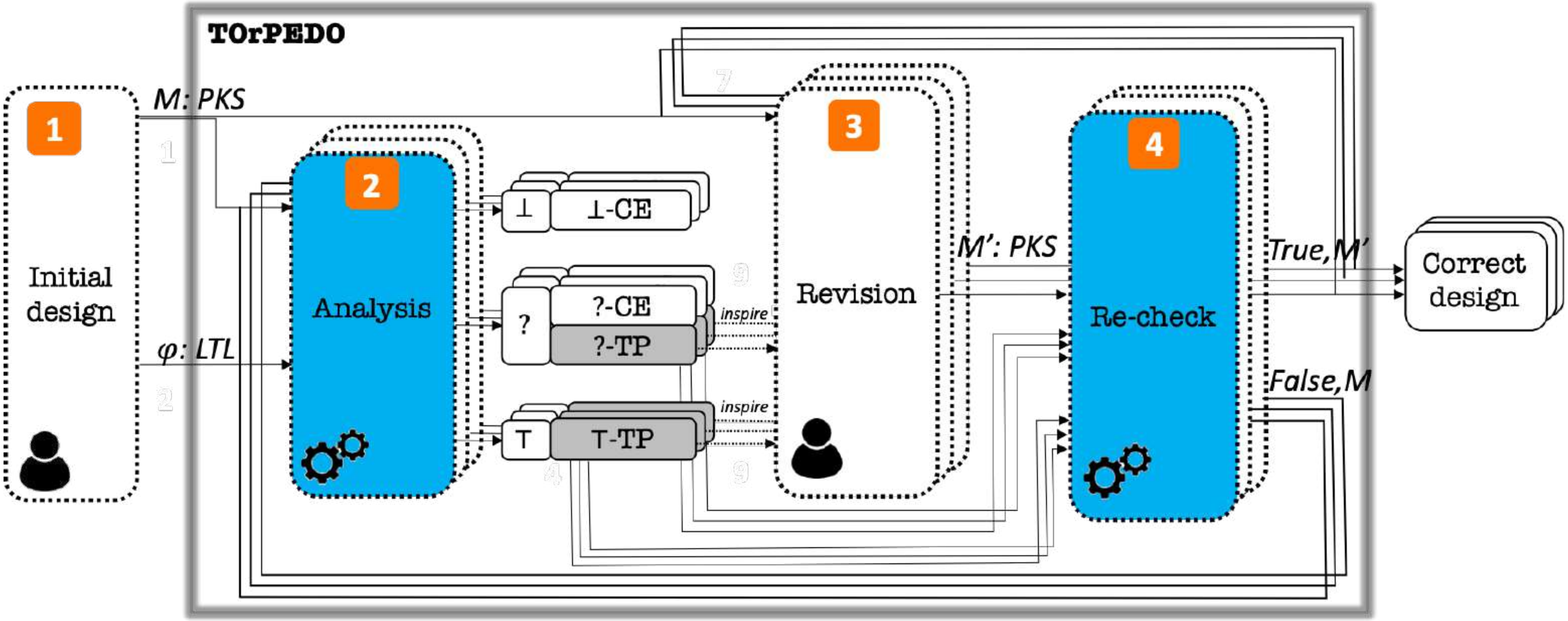

TOrPEDO
1
Initial design
M: PKS
φ: LTL
2
Analysis
⊥
⊥-CE
?
?-CE
?-TP
T
T-TP
inspire
inspire
3
Revision
M': PKS
4
Re-check
True,M'
False,M
Correct design
part with relation

# TOrPEDO



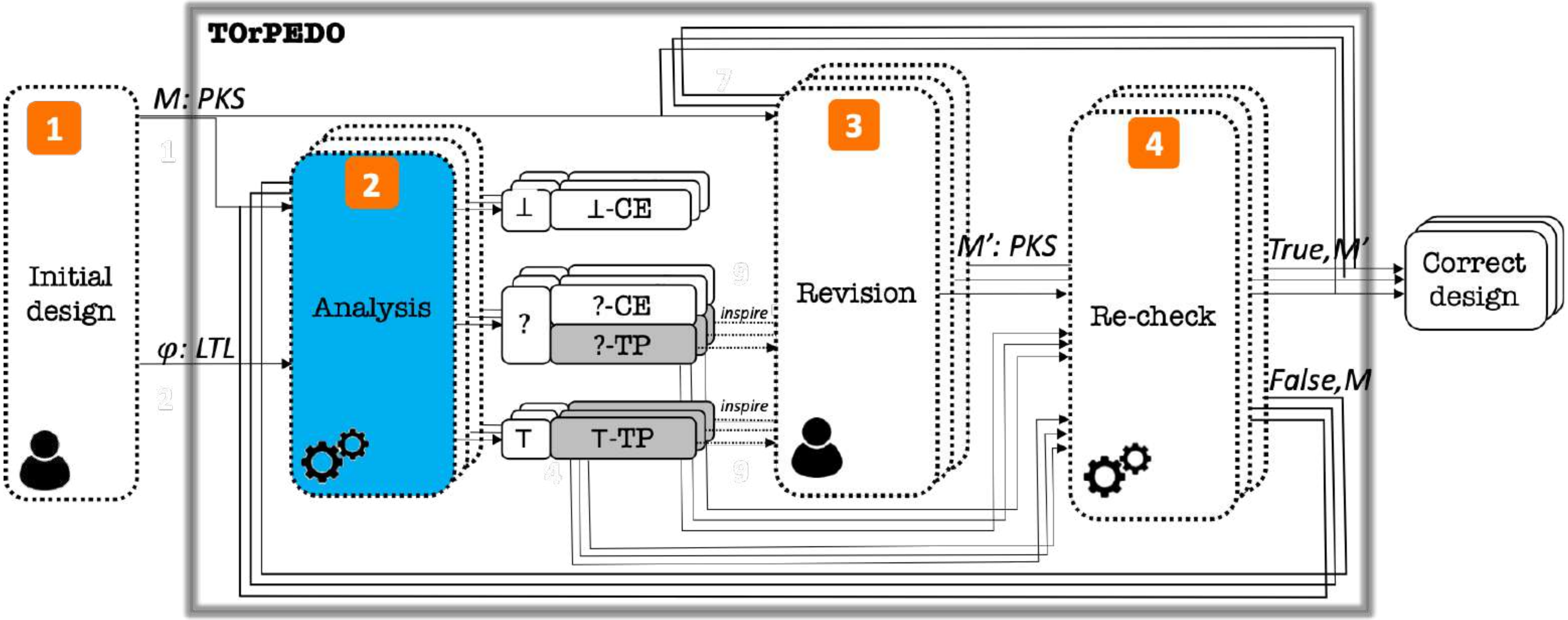
TOrPEDO
1
Initial design
M: PKS
φ: LTL
2
Analysis
⊥
⊥-CE
?
?-CE
?-TP
T
T-TP
inspire
inspire
3
Revision
M': PKS
4
Re-check
True,M'
False,M
Correct design

# TOrPEDO



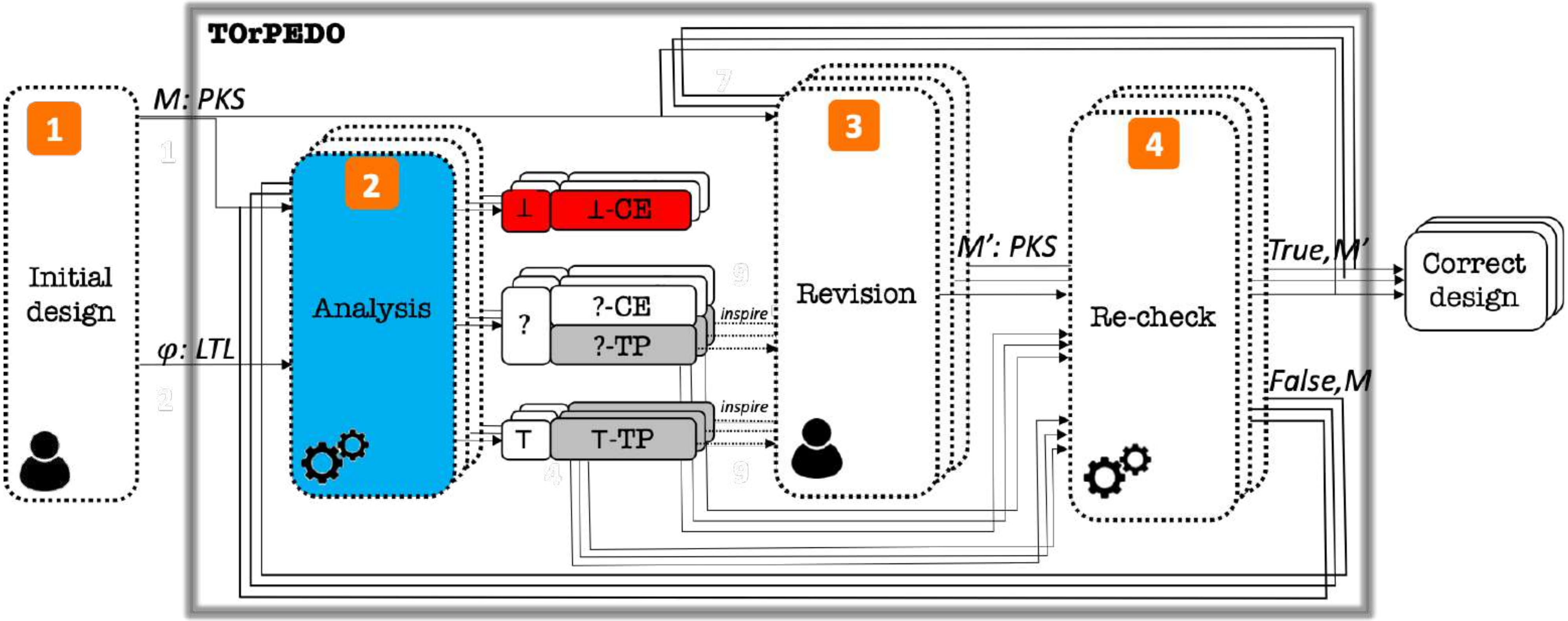
TOrPEDO
M: PKS
1
Initial design
φ: LTL
2
Analysis
⊥
⊥-CE
?
?-CE
?-TP
T
T-TP
inspire
inspire
3
Revision
M': PKS
4
Re-check
True,M'
False,M
Correct design

# TOrPEDO



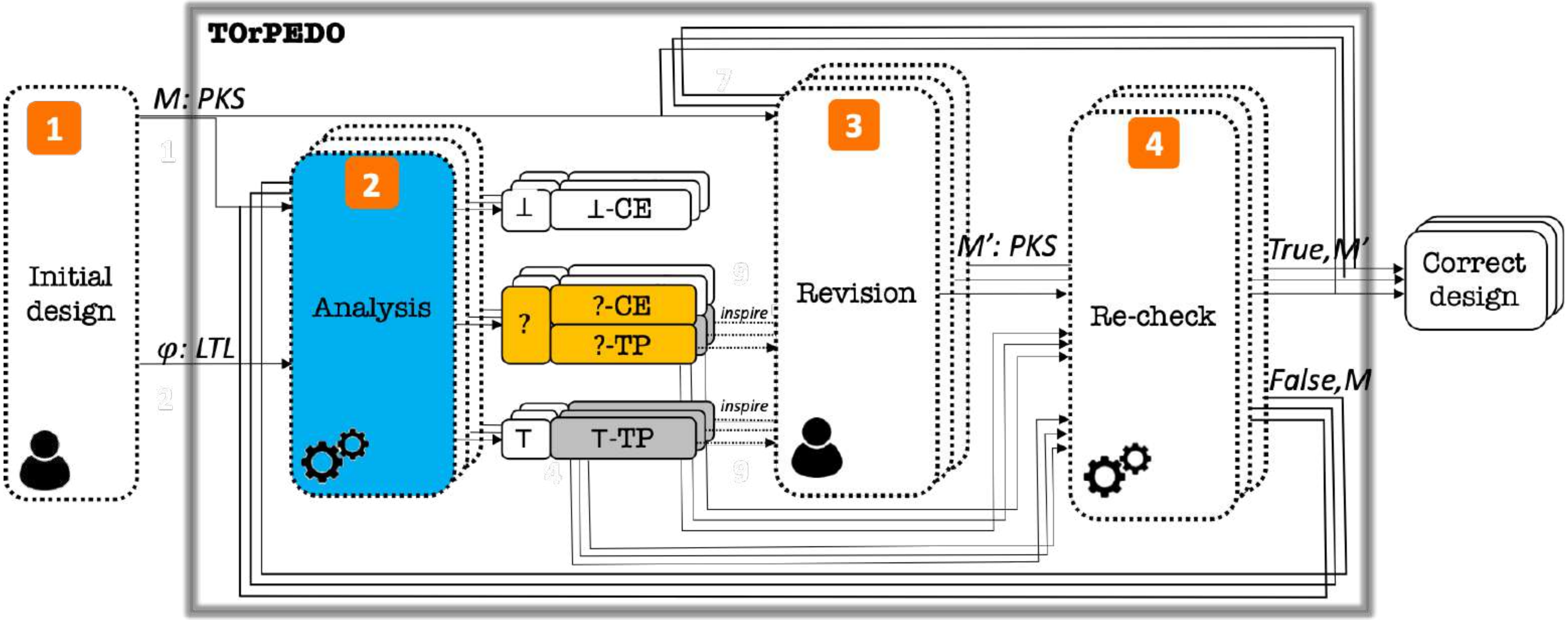
TOrPEDO
M: PKS
1
Initial design
φ: LTL
2
Analysis
⊥
⊥-CE
?
?-CE
?-TP
T
T-TP
inspire
inspire
3
Revision
M': PKS
4
Re-check
True,M'
False,M
Correct design

# TOrPEDO



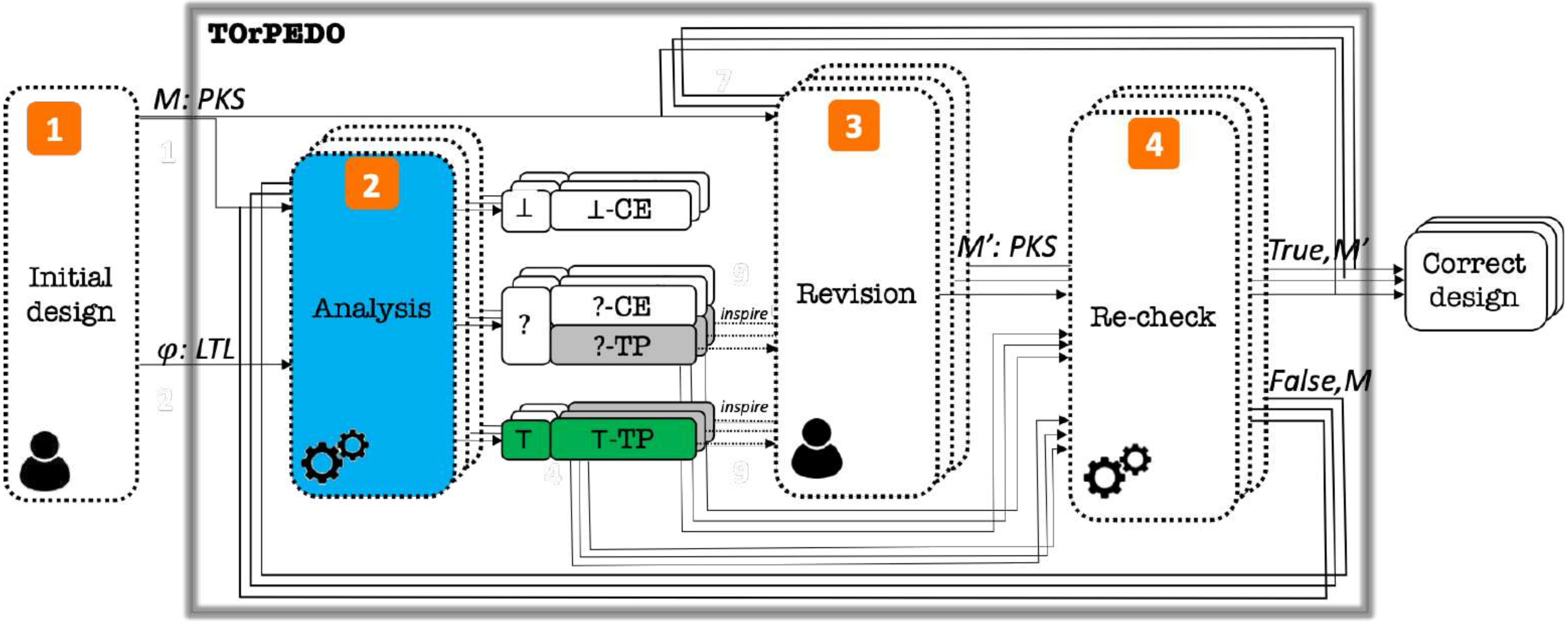
TOrPEDO
M: PKS
1
Initial design
φ: LTL
2
Analysis
⊥
⊥-CE
?
?-CE
?-TP
T
T-TP
inspire
inspire
3
Revision
M': PKS
4
Re-check
True,M'
False,M
Correct design

# TOrPEDO



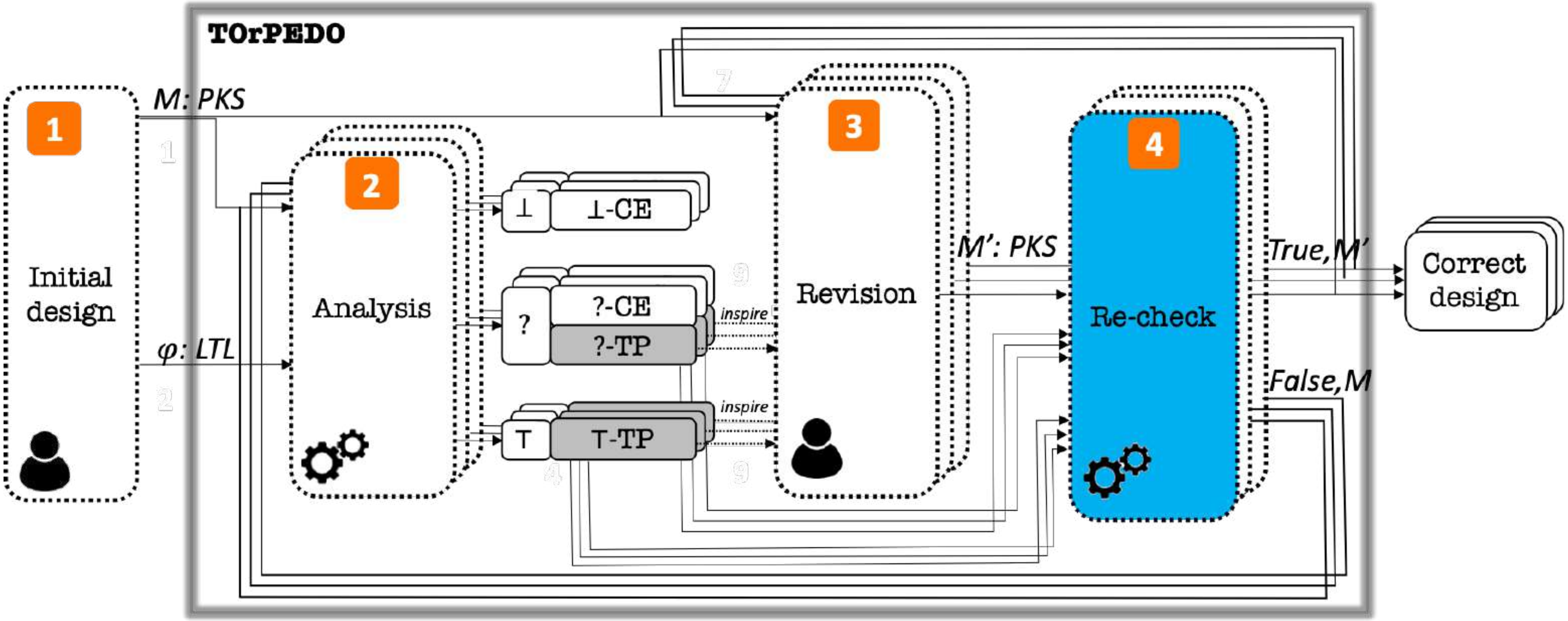
TOrPEDO
1
Initial design
M: PKS
φ: LTL
2
Analysis
⊥
⊥-CE
?
?-CE
?-TP
T
T-TP
inspire
inspire
3
Revision
M': PKS
4
Re-check
True,M'
False,M
Correct design

# TOrPEDO



TOrPEDO
M: PKS
1
Initial design
φ: LTL
2
Analysis
⊥
⊥-CE
?
?-CE
?-TP
T
T-TP
inspire
inspire
3
Revision
M': PKS
4
Re-check
True, M'
False, M
Correct design

# TOrPEDO



TOrPEDO
M: PKS
1
Initial design
φ: LTL
2
Analysis
⊥
⊥-CE
?
?-CE
?-TP
T
T-TP
inspire
inspire
3
Revision
M': PKS
4
Re-check
True,M'
False,M
Correct design

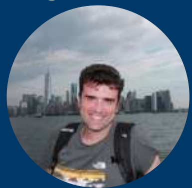
# Topological Proofs

## Integrating Topological Proofs with Model Checking to Instrument Iterative Design

Fundamental Approaches to Software Engineering (FASE)

2020


Menghi, Claudio



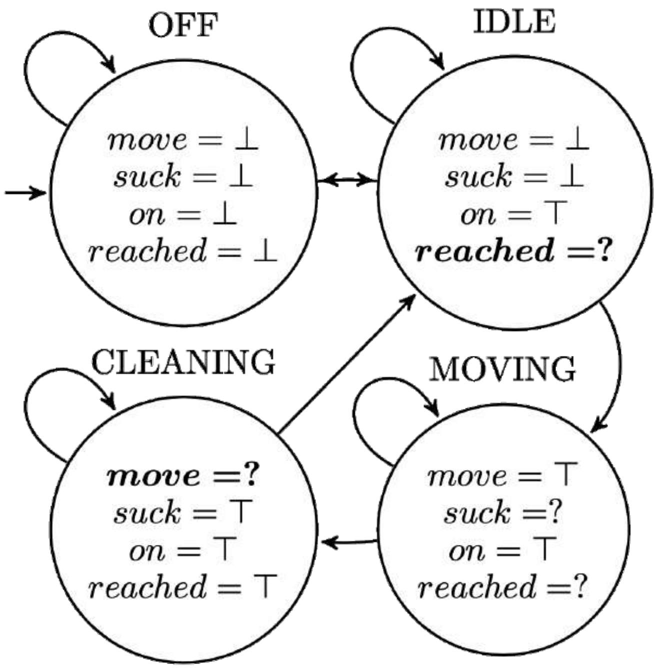
Rizzi, Alessandro Maria

Bernasconi, Anna

# Topological Proofs

A topological proof is a slice of the model that witnesses property satisfaction

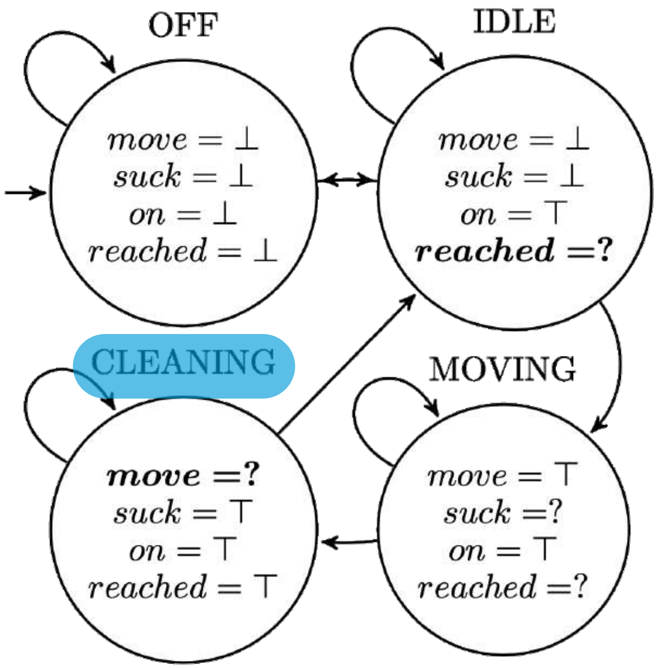
# Topological Proofs




**Propositional Clause (TPP)**

⟨CLEANING, reached, ⊤⟩

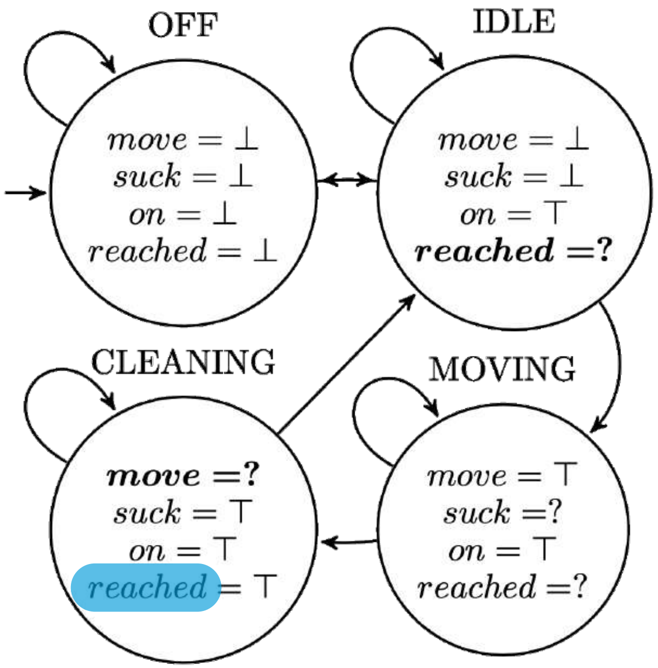
# Topological Proofs




**Propositional Clause (TPP)**

⟨CLEANING, reached, ⊤⟩

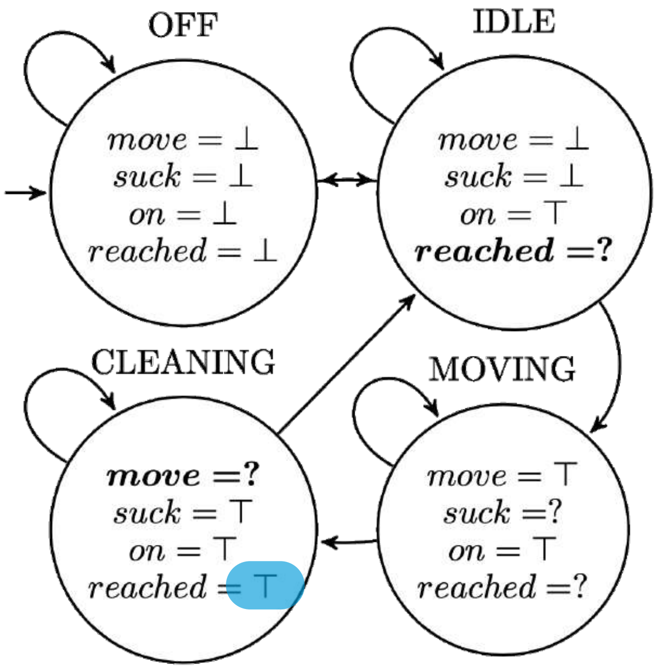
# Topological Proofs




**Propositional Clause (TPP)**

$\langle$CLEANING, reached, $\top\rangle$

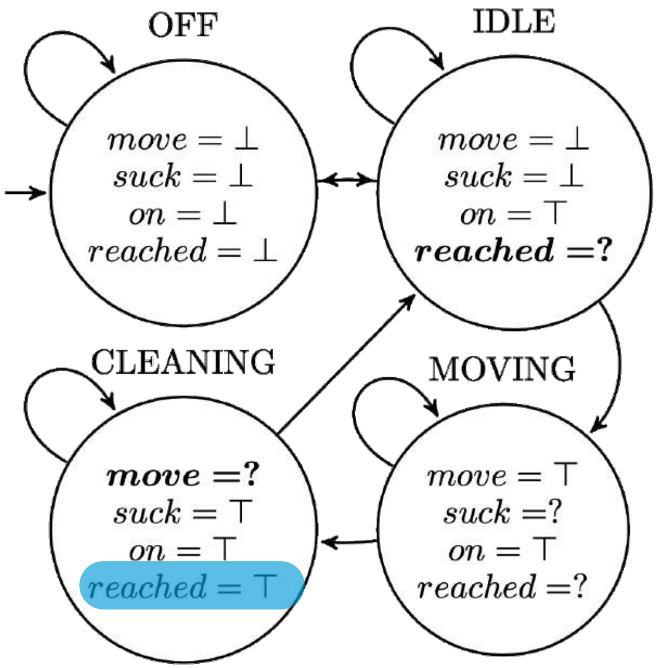
# Topological Proofs




**Propositional Clause (TPP)**

⟨CLEANING, reached, ⊤⟩

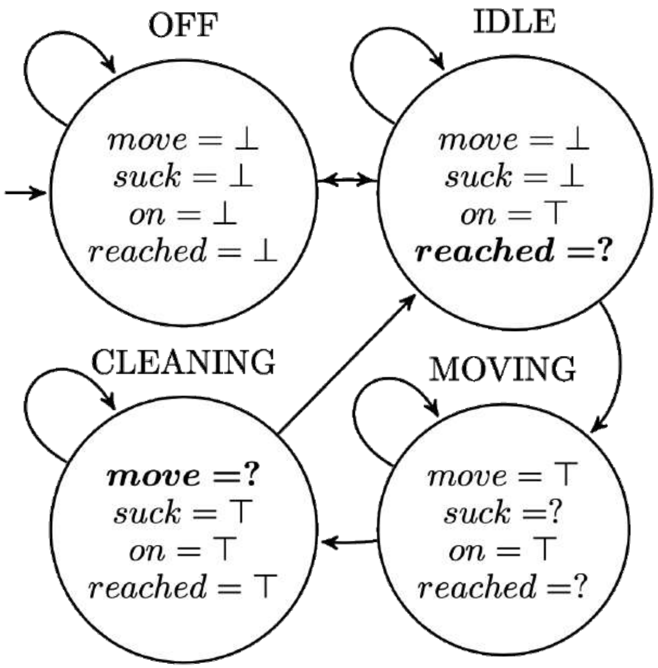
# Topological Proofs




**Propositional Clause (TPP)**

⟨CLEANING, reached, ⊤⟩

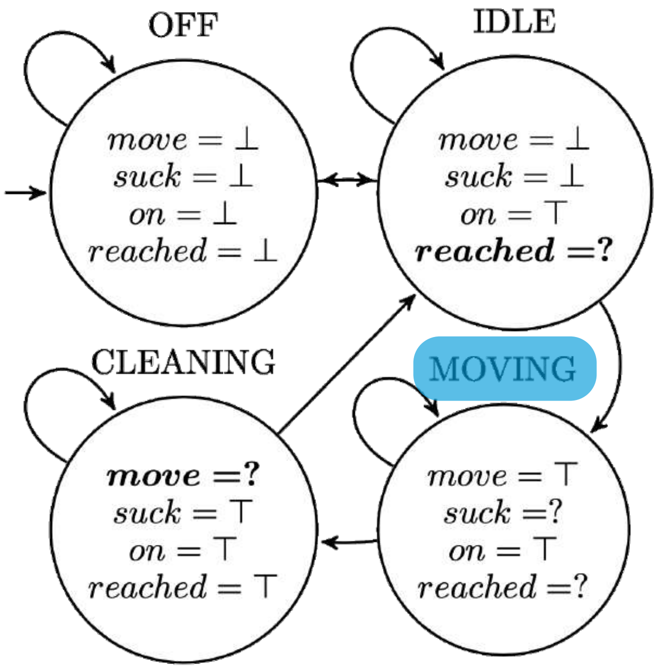
# Topological Proofs




**Transitions-from-state Clause (TPT)**

⟨MOVING,{MOVING,CLEANING}⟩

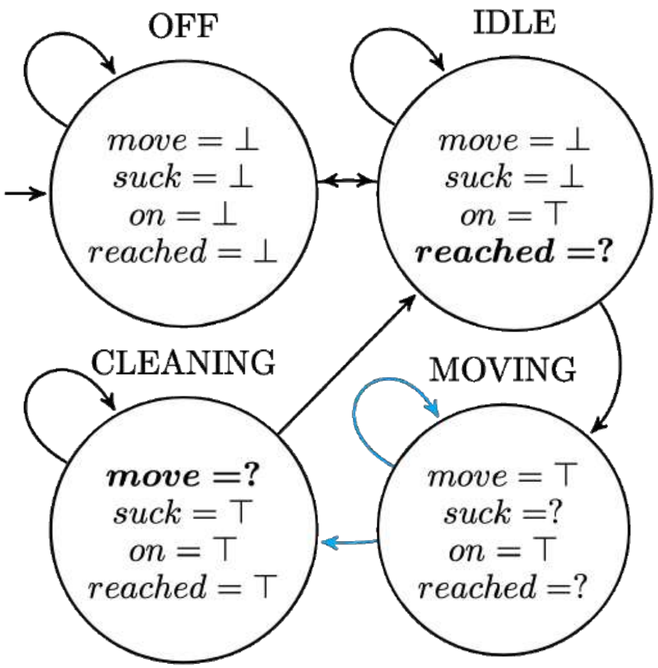
# Topological Proofs




**Transitions-from-state Clause (TPT)**

⟨MOVING,{MOVING,CLEANING}⟩

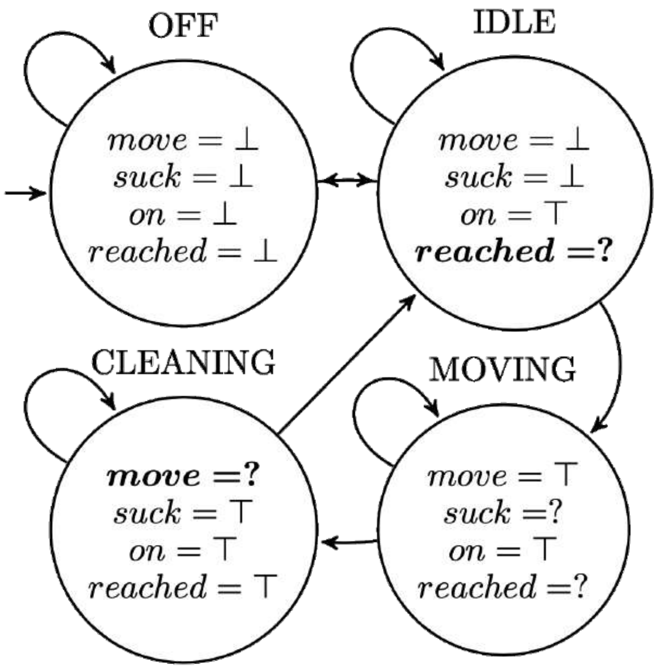
# Topological Proofs




## Transitions-from-state Clause (TPT)

⟨MOVING, {MOVING, CLEANING}⟩

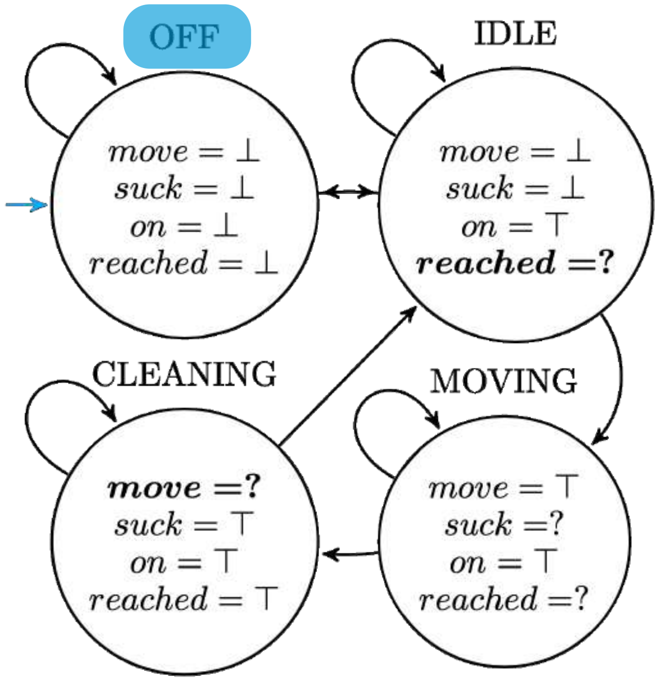
# Topological Proofs




**Initial-states Clause (TPI)**

⟨{OFF}⟩

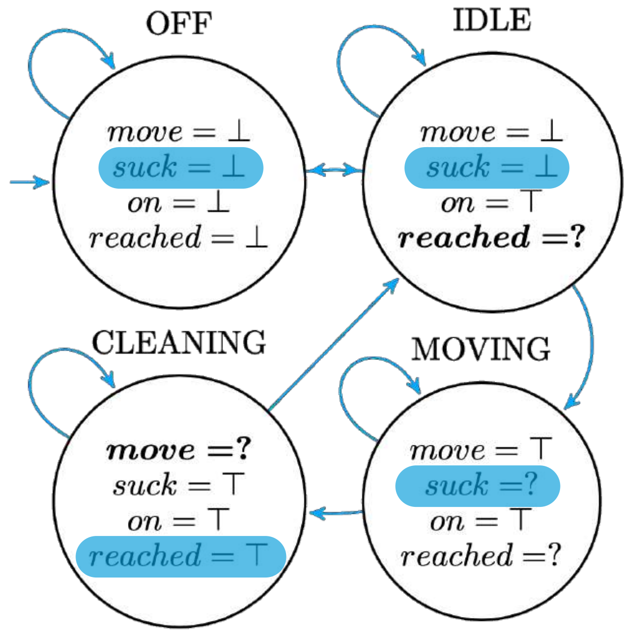
# Topological Proofs




## Initial-states Clause (TPI)

⟨{OFF}⟩

# Topological Proofs




**TPP**: $\langle CLEANING, reached, \top \rangle$ $\langle OFF, suck, \bot \rangle$, $\langle IDLE, suck, \bot \rangle$, $\langle MOVING, suck, ?\ \rangle$
**TPT**: $\langle OFF, \{OFF, IDLE\} \rangle$, $\langle IDLE, \{OFF, IDLE, MOVING\} \rangle$,
$\langle MOVING, \{MOVING, CLEANING\} \rangle$, $\langle CLEANING, \{CLEANING, IDLE\} \rangle$
**TPI**: $\langle \{OFF\} \rangle$

$\phi_1$: the robot is drawing dust (*suck*) only if it has *reached* the cleaning site. $\phi_1 \equiv \mathcal{G}(suck \rightarrow reached)$

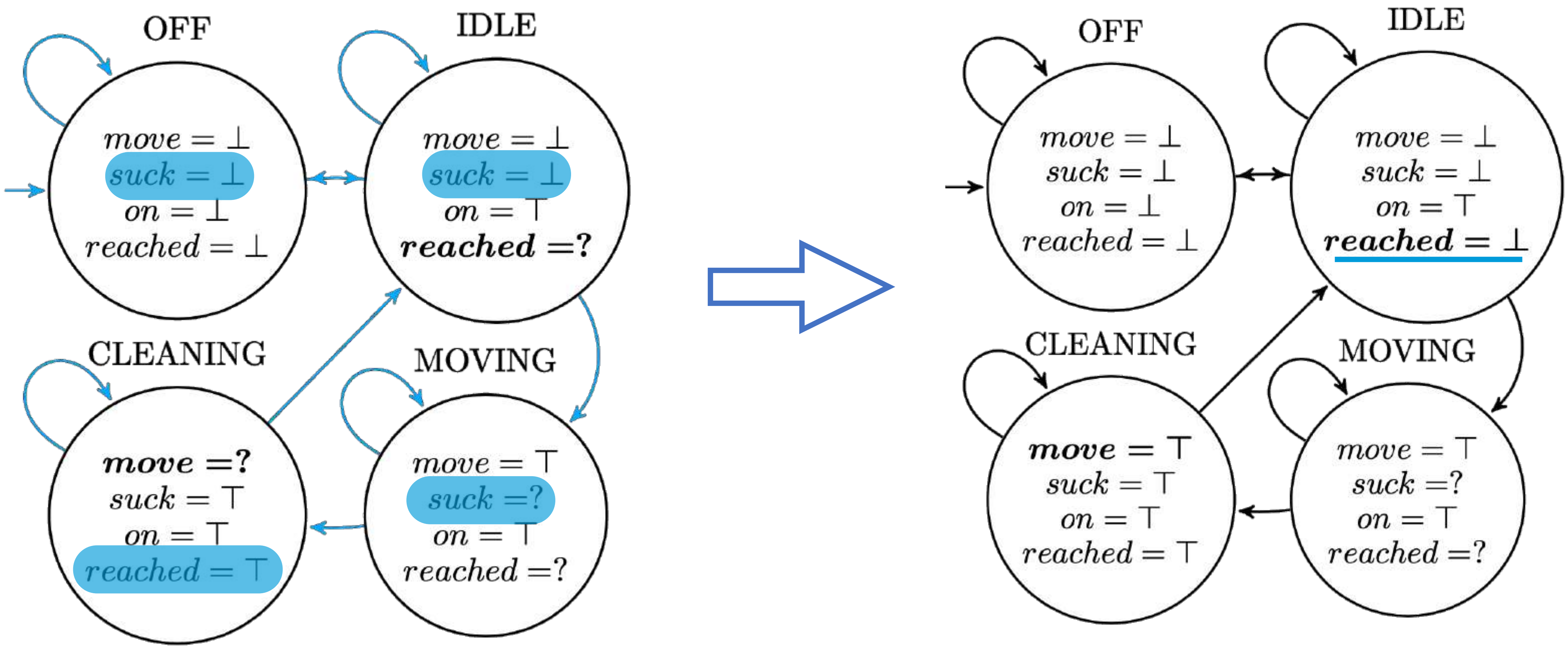
## Topological Proofs

- **Revision rules**. An engineer should not
  - add or remove transitions whose source state is in a transition included in the TPT-clauses;
  - change the value of propositions that are in a TPP-clause;
  - remove states that are in any TPT, TPP, or TPI clause;
  - change the initial states if they are in a TPI-clause.

# Topological Proofs

If the engineer follows the revision rules,
then the revision will not violate the property

# Vacuum-cleaner robot: Revision



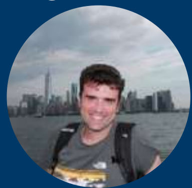
# Automated Support

## Integrating Topological Proofs with Model Checking to Instrument Iterative Design

Fundamental Approaches to Software Engineering (FASE)

2020

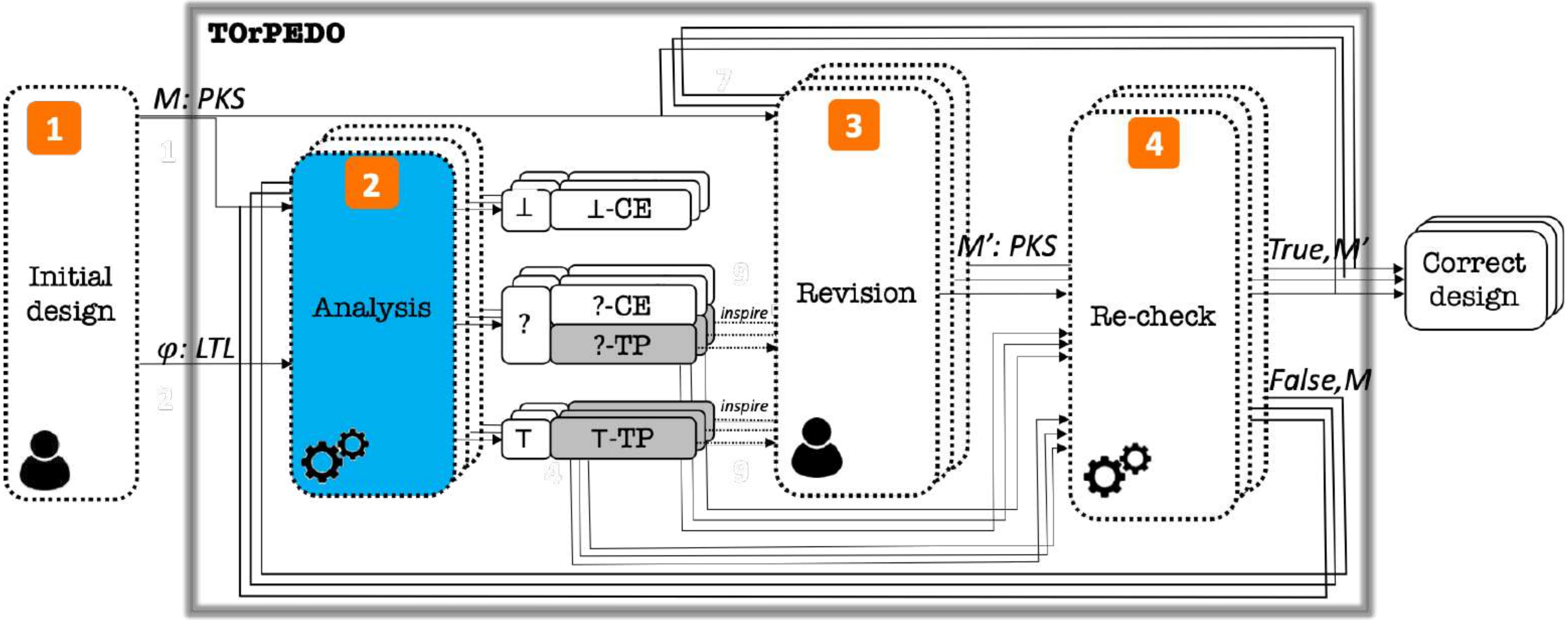

Menghi, Claudio



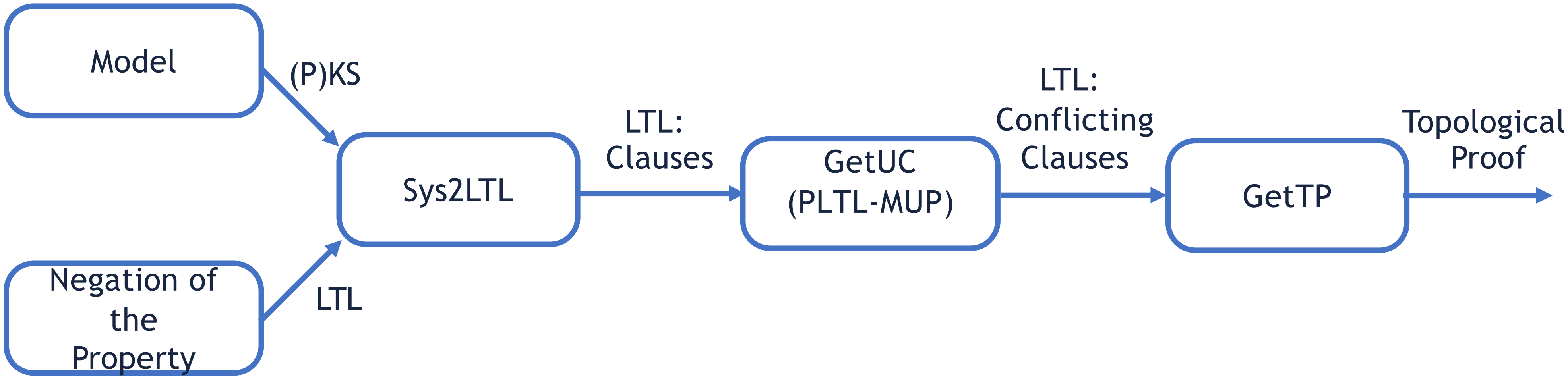
Rizzi, Alessandro Maria

Bernasconi, Anna

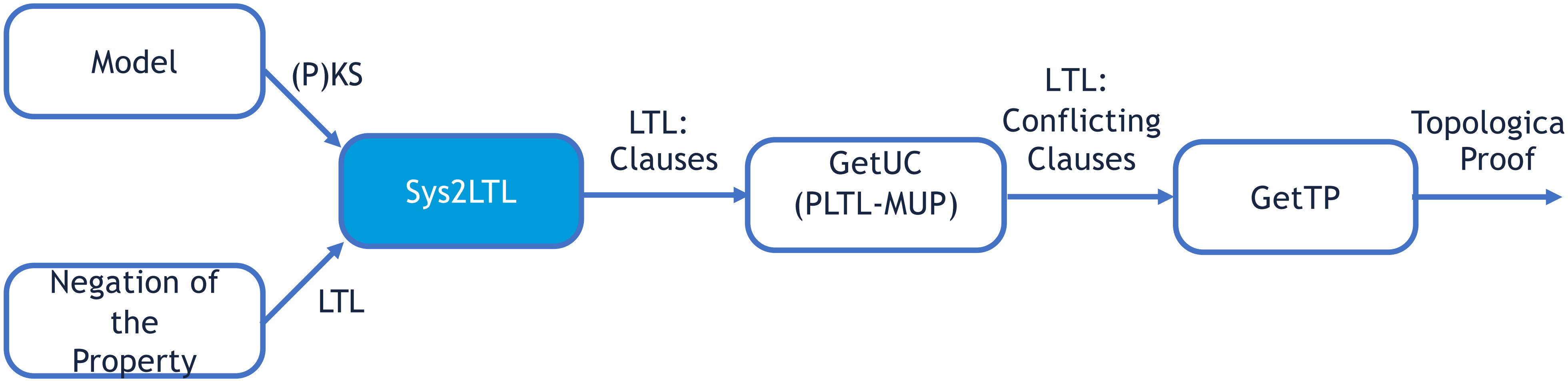
# Automated Support



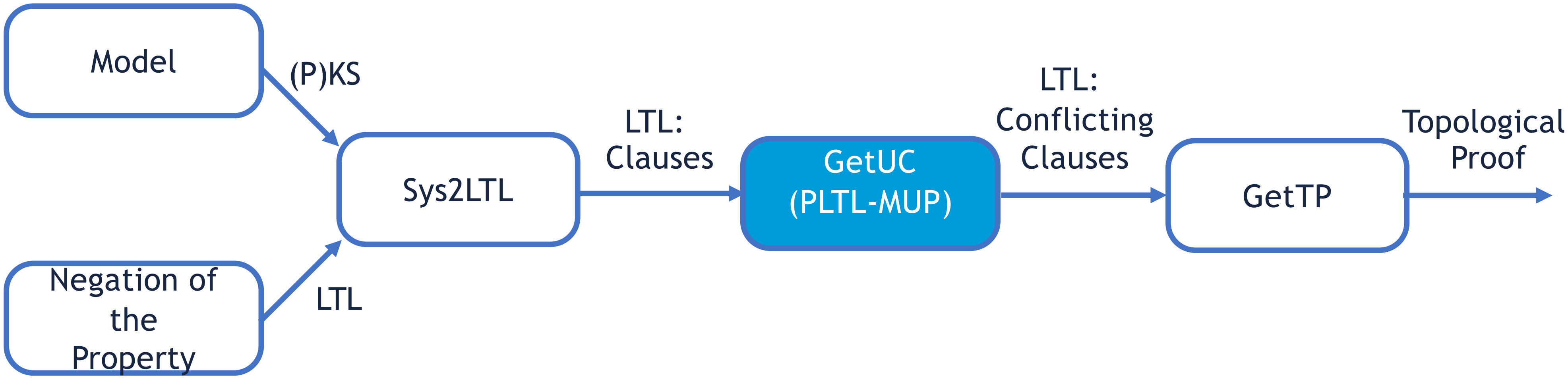
# Topological proof computation



# Topological proof computation



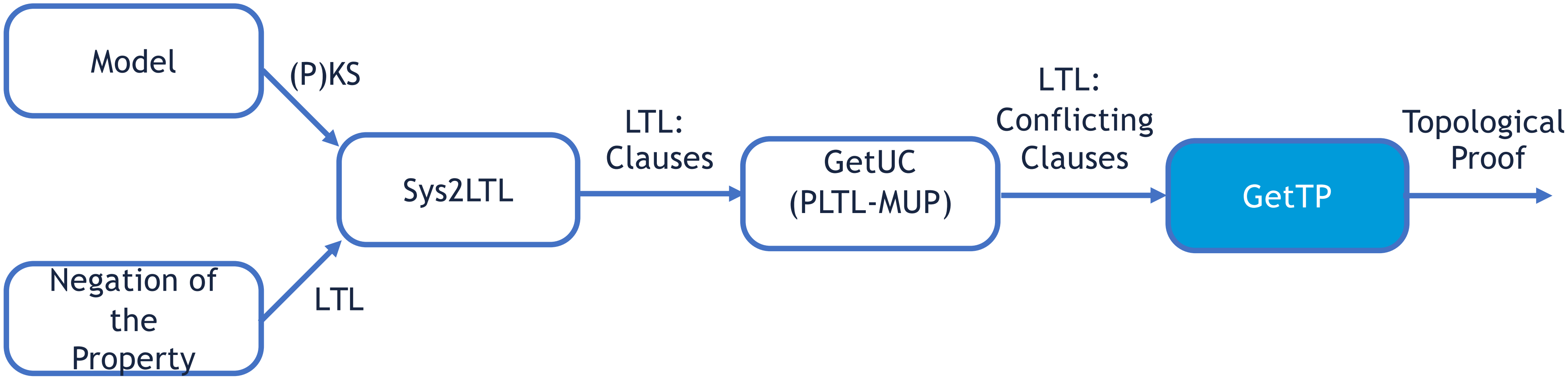
# Topological Proof Computation



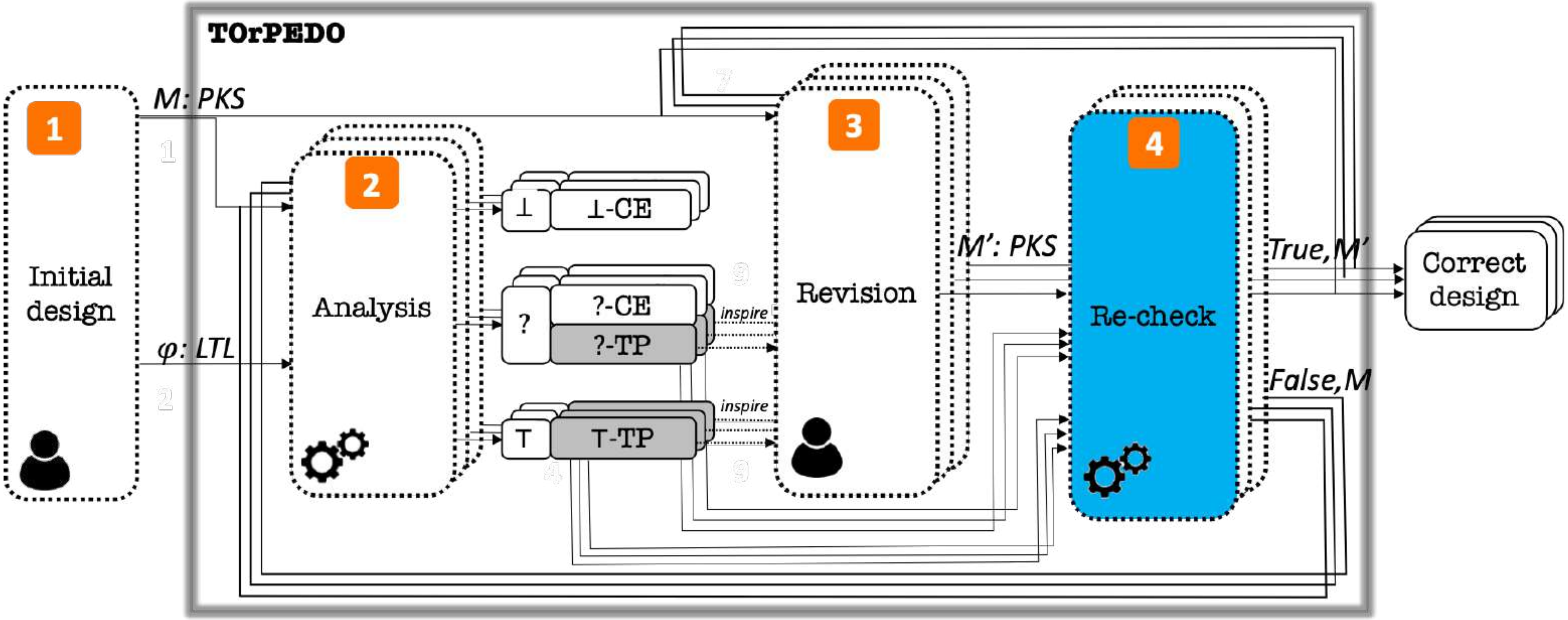

* In our experiments we considered an extended version of PLTL-MUP, namely Hybrid, that improves the PLTL-MUP performances by combining it with TRP++UC.

Finding minimal unsatisfiable subsets in linear temporal logic using BDDs,
Sergeant T, Goré SR, Thomson J (2013)
https://cs. anu.edu.au/courses/csprojects/13S1/Reports/Timothy_Sergeant_Report.pdf .

# Topological proof computation



# Re-check


TOrPEDO
M: PKS
1
Initial design
φ: LTL
2
Analysis
⊥
⊥-CE
?
?-CE
?-TP
T
T-TP
inspire
inspire
3
Revision
M': PKS
4
Re-check
True,M'
False,M
Correct design

# Re-check

The re-check verifies that the engineer did not:

- add or remove transitions whose source state is in a transition included in the TPT-clauses;
- change the value of propositions that are in a TPP-clause;
- remove states that are in any TPT, TPP, or TPI clause;
- change the initial states if they are in a TPI-clause.

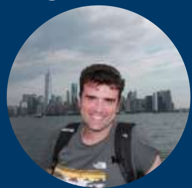
# Evaluation

## Integrating Topological Proofs with Model Checking to Instrument Iterative Design

Fundamental Approaches to Software Engineering (FASE)

2020


Menghi, Claudio



Rizzi, Alessandro Maria

Bernasconi, Anna

# Evaluation

- **RQ1**: How does the size of the proofs computed by the **analysis** component compares with the size of the original models?

# RQ1: Size of The Topological Proofs

- We considered 60 model-requirement combinations
  - 12 models (PKS)
  - five properties per model
- We run TOrPEDO and computed the topological proofs
- We compared the size of the topological proof and the size of the model

# RQ1: Size of The Topological Proofs

Topological proofs are approximately 60% smaller than the respective models

# Evaluation

- **RQ1**: How does the size of the proofs computed by the **analysis** component compares with the size of the original models?
- **RQ2**: How does the **re-check** component support the creation of model revisions?

## RQ2: Support Provided by the Re-check Component

- We considered three models and five properties per model
  - for each model we considered four revisions
- We run TOrPEDO and computed the topological proofs
- We computed the percentage of cases in which the re-check component confirmed that the revision was compliant with the topological proof

# RQ2: Support Provided by the Re-check Component

In 78% of the cases, the re-check component confirmed that the revision was compliant with the topological proof.

# Evaluation

- **RQ1**: How does the size of the proofs computed by the **analysis** component compares with the size of the original models?
- **RQ2**: How does the **re-check** component support the creation of model revisions?
- **RQ3**: What is the **scalability** of TOrPEDO?

# RQ3: Scalability of TOrPEDO

- To have a ballpark estimation of the scalability of TOrPEDO we
  - assessed its performance on the models used in RQ1 and RQ2
  - manually designed an additional model with 10 states and 5 atomic propositions and 26 transitions

# RQ3: Scalability of TOrPEDO

For the models of RQ1 and RQ2, TOrPEDO required on average less than 10s to compute the topological proof.

For the additional example, the topological proof was computed in 1m33s.

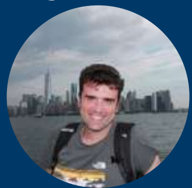
# Conclusions

## Integrating Topological Proofs with Model Checking to Instrument Iterative Design

Fundamental Approaches to Software Engineering (FASE)

2020


Menghi, Claudio



Rizzi, Alessandro Maria

Bernasconi, Anna

UNIVERSITÀ DEGLI STUDI DI BERGAMO

Dipartimento di Ingegneria Gestionale, dell'Informazione e della Produzione

# Conclusions and Future Work

- We proposed TOrPEDO, an integrated framework that supports the iterative model design
- We defined the novel notion of Topological Proofs
- We evaluated TOrPEDO by assessing the support provided by the analysis and re-check components and their scalability

# Conclusions and Future Work

Our results show that

- proofs are 60% smaller than the original models
- revision can be verified 78% of the cases by executing a simple syntactic check
- the scalability of existing tools is not sufficient

# Conclusions and Future Work

**Future Work**: We need to develop a more efficient procedure to extract topological proofs

2021

# TOrPEDO: Witnessing Model Correctness with Topological Proofs

Formal Aspects of Computing (FAOC)


Menghi, Claudio

Rizzi, Alessandro Maria

Bernasconi, Anna

Spoletini, Paola

## 2021

# TOrPEDO: Witnessing Model Correctness with Topological Proofs

Formal Aspects of Computing (FAOC)


Menghi, Claudio

Rizzi, Alessandro Maria

Bernasconi, Anna

Spoletini, Paola


# Introduction

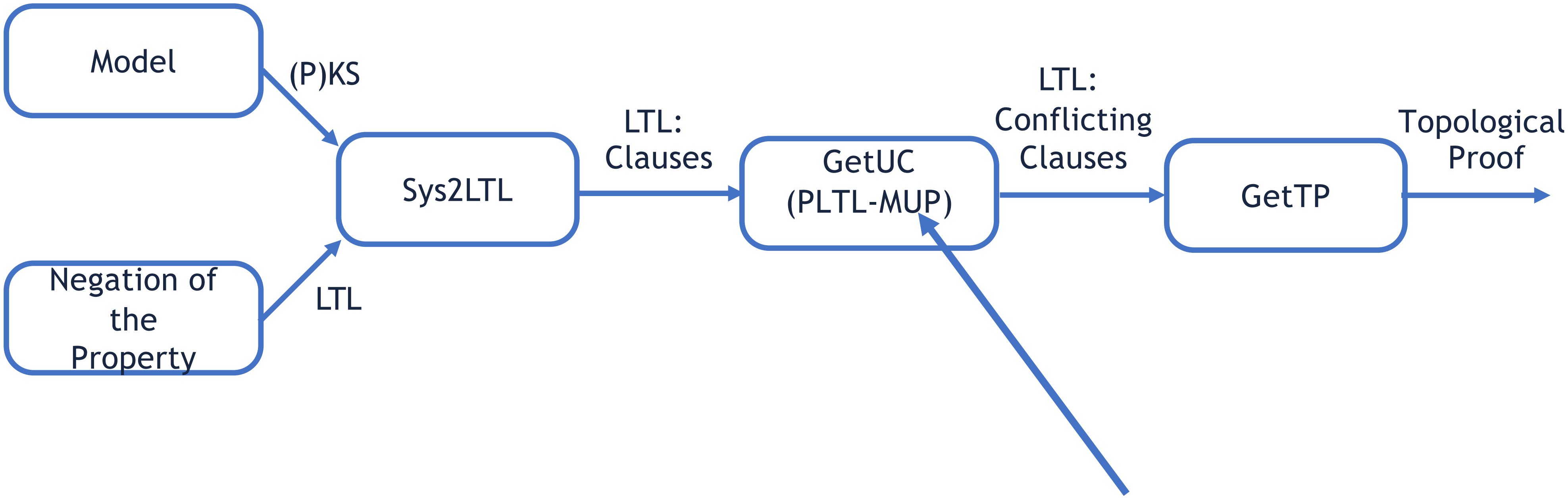
## Problem Definition

In our previous work, we implemented TOrPEDO using

- NuSMV as a model checker, and
- PLTL-MUP to compute a minimal subset of unsatisfiable LTL formulae (from an unsatisfiable set of LTL formulae)

We will refer to this instance of TOrPEDO as TOrPEDO-MUP.

# Topological Proof Computation



# Problem

Can we reduce the computational cost required to compute topological proofs?

2021

## TOrPEDO: Witnessing Model Correctness with Topological Proofs

Formal Aspects of Computing (FAOC)

Menghi, Claudio

Rizzi, Alessandro Maria

Bernasconi, Anna

Spoletini, Paola

# Contribution

# Contribution: TOrPEDO-SMT

We propose TOrPEDO-SMT

- converts LTL formulae into an SMT problem*

* Linear encodings of bounded LTL model checking
Schuppan V, Latvala T, Junttila T, Heljanko K, Biere A (2006)
Log Methods Comput Sci, 2,
Episciences.org

# Contribution: TOrPEDO-SMT

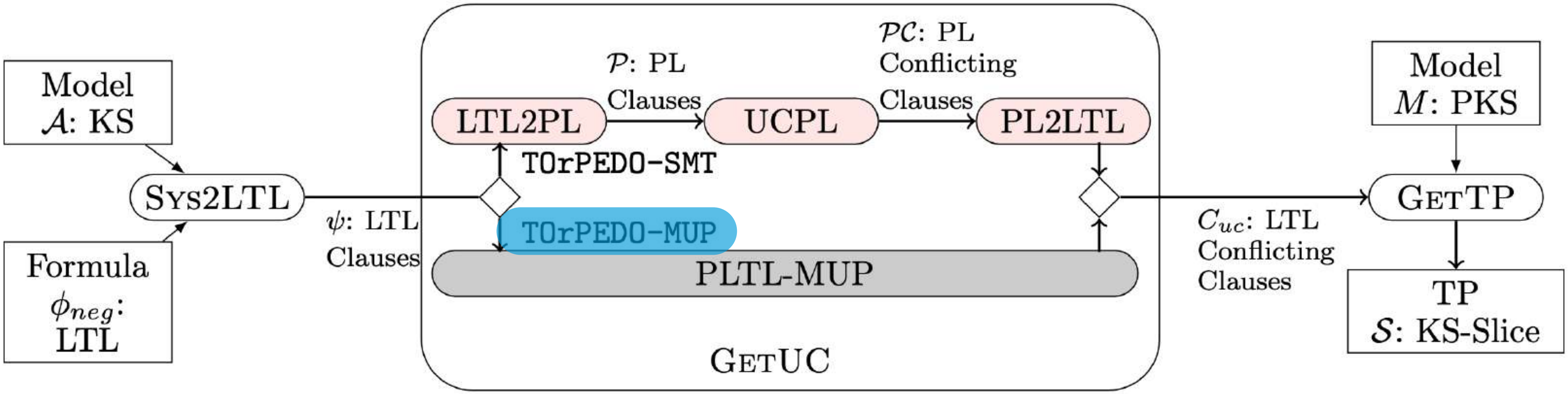
We propose TOrPEDO-SMT

- converts LTL formulae into an SMT problem*
- relies on Bit-Vectors**

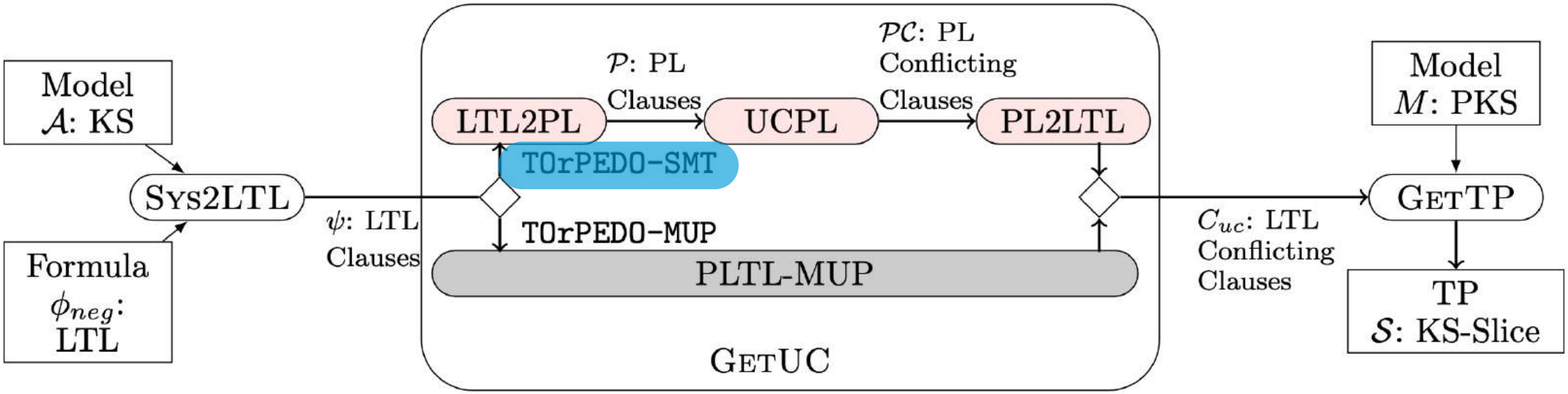
** Efficient scalable verification of LTL specifications
Baresi L, Kallehbasti MMP, Rossi M (2015)
International conference on software engineering, pp 711–721. IEEE

** On how bit-vector logic can help verify LTL-based specifications.
Pourhashem KMM, Rossi MG, Baresi L (2020)
IEEE Trans Softw Eng, pp 1–1

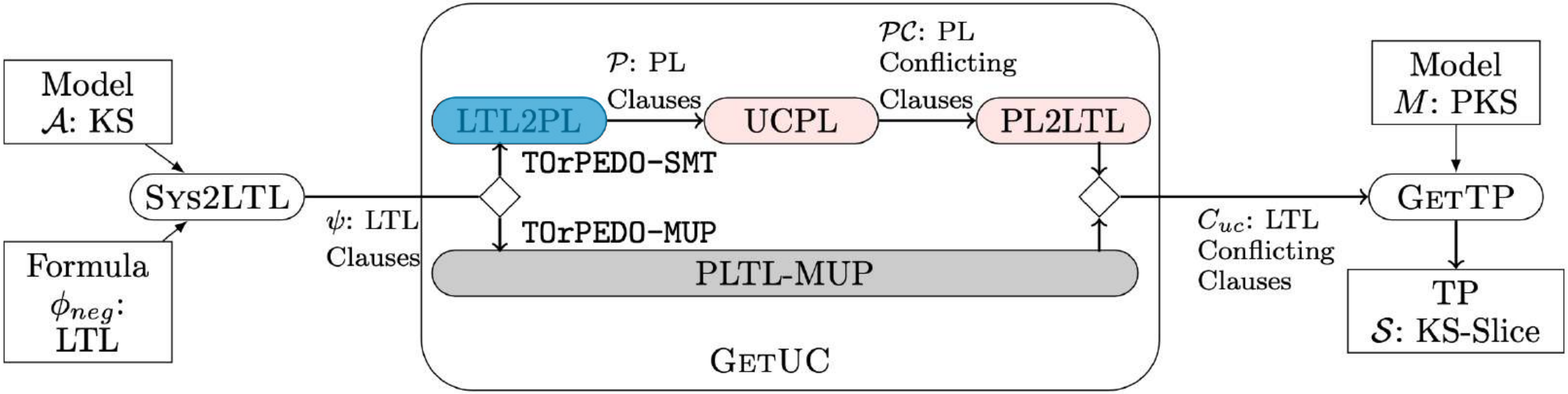
# Contribution: TOrPEDO-SMT



# Contribution: TOrPEDO-SMT



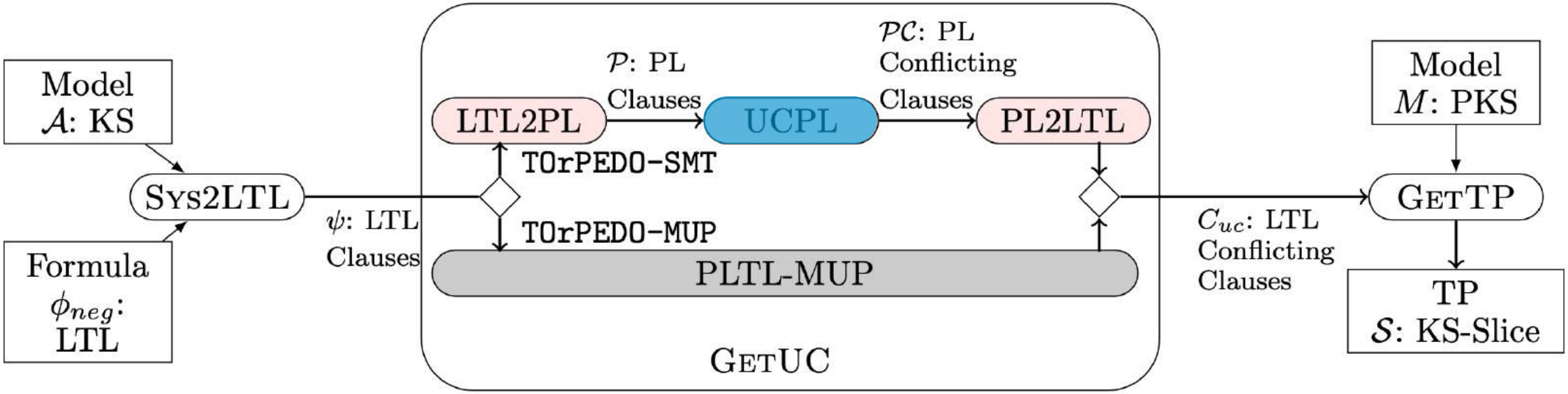
# Contribution: TOrPEDO-SMT




**LTL2PL**: converts LTL formulae into PL (Propositional Logic)

- Unrolls the LTL formula up to length k

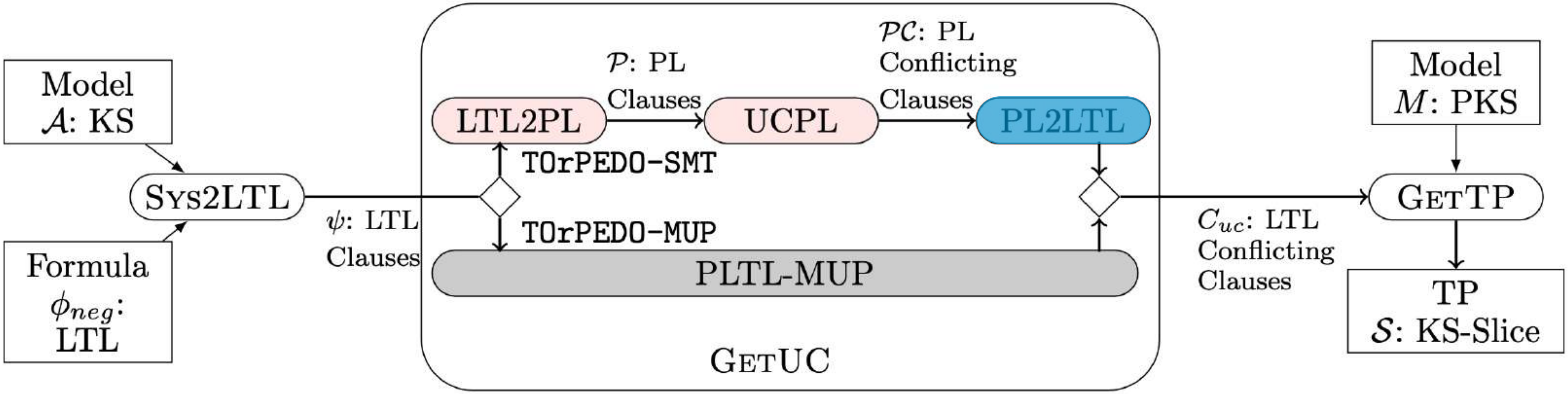
# Contribution: TOrPEDO-SMT




**GetUC**: computes the unsatisfiable core of a PL formula

- we employ the Z3 Theorem Prover

# Contribution: TOrPEDO-SMT




**PL2LTL**: maps the conflicting propositional clauses to LTL

2021

# TOrPEDO: Witnessing Model Correctness with Topological Proofs

Formal Aspects of Computing (FAOC)


Menghi, Claudio

Rizzi, Alessandro Maria

Bernasconi, Anna

Spoletini, Paola


## Evaluation

# Evaluation

- **RQ3**: How efficient is TOrPEDO in analyzing models and how does TOrPEDO-SMT compare to TOrPEDO-MUP?

# Comparison of Efficiency (RQ3): Benchmark

- We generated a set of random models
- The models have an increasing number of states (i.e., 10, 20, 30, and 40)
- The models are generated from the grade crossing semaphore example
- We considered two properties (satisfied and possibly satisfied)

# Comparison of Efficiency (RQ3): Methodology

- We run TOrPEDO-MUP and TOrPEDO-SMT
- For TOrPEDO-SMT, we set 86 for the bound k*
- We set two hours as the timeout

* We selected this value since it ensures the correctness of the result, i.e., we set its value by considering to the size of the recurrence diameter (the longest initialised loop-free path in the state graph) and the size of the Büchi automaton representing the negation of the property

Clarke E, Kroening D, Ouaknine J, Strichman O (2005)
Computational challenges in bounded model checking.
Int J Softw Tools Technol Transf 7(2):174–183

# Comparison of Efficiency (RQ3): Results

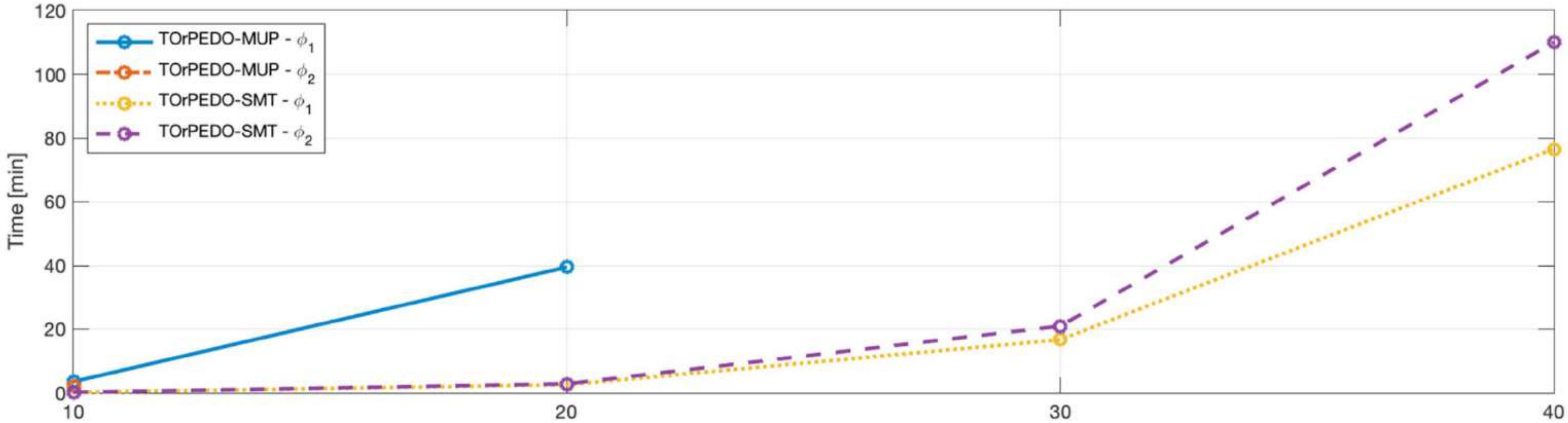



**Fig. 4.** Comparison of the efficiency of `TOrPEDO-MUP` and `TOrPEDO-SMT`. For the property $\phi_2$, `TOrPEDO-MUP` provided a result only for the model with 10 states in 2.1m

# Comparison of Efficiency (RQ3): Results

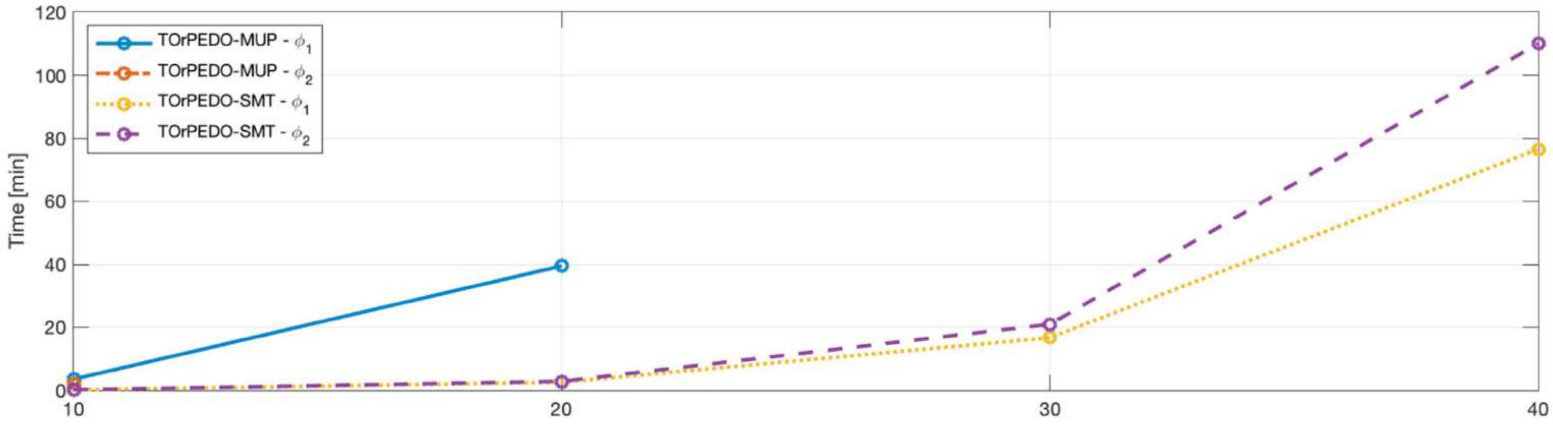



**Fig. 4.** Comparison of the efficiency of `TOrPEDO-MUP` and `TOrPEDO-SMT`. For the property $\phi_2$, `TOrPEDO-MUP` provided a result only for the model with 10 states in 2.1m

The answer to RQ3 is that, on the considered models,

- TOrPEDO-SMT can verify within the timeout models which are double in size compared to TOrPEDO-MUP

# Comparison of Efficiency (RQ3): Results

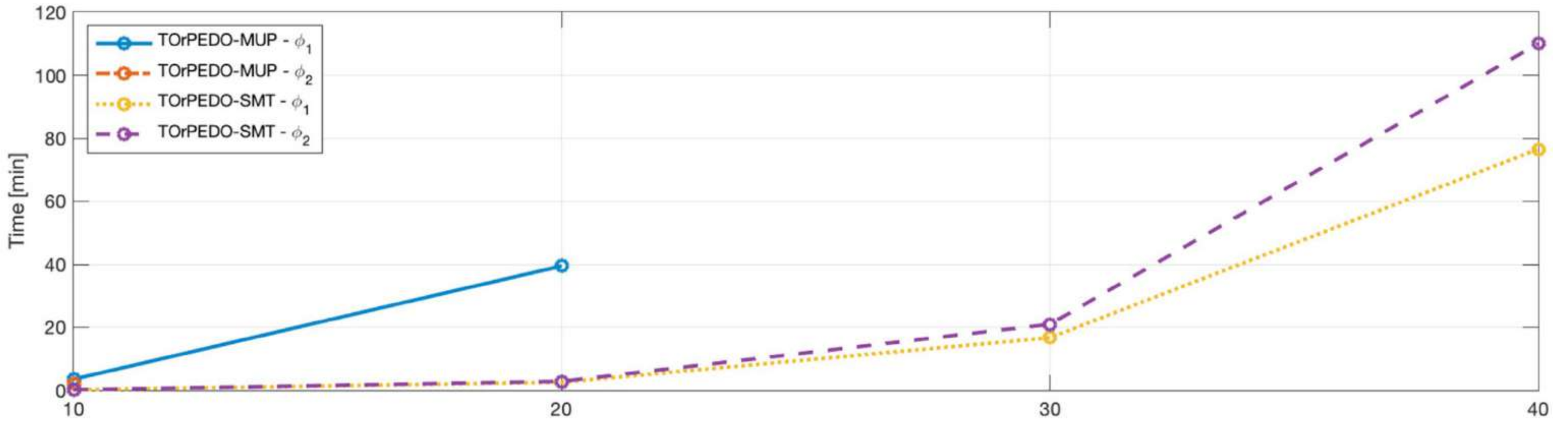



**Fig. 4.** Comparison of the efficiency of TOrPEDO-MUP and TOrPEDO-SMT. For the property $\phi_2$, TOrPEDO-MUP provided a result only for the mod...

When both tools finished within the timeout, TOrPEDO-SMT is significantly faster than TOrPEDO-MUP.
TOrPEDO-SMT required on average 1.4m, TOrPEDO-MUP required 15m.

# Evaluation

- **RQ4**: How useful is TOrPEDO-SMT in supporting the designers in the model design on an example in the genomic domain?

# Usefulness (RQ4): Benchmark Model

- We considered a (small) model from the genomic domain, related to Gene Regulatory Networks (GRNs).
- GRNs are collections of molecular regulators, interacting with each other

# Usefulness (RQ4): Benchmark Model

- The PKS represents the status of genes with propositions
- The proposition is true if the gene is activated.
- states describes the status of the genes
- The PKS consists of 64 states
- Transitions encode how the status of the genes can change

# Usefulness (RQ4): Benchmark Model

- We considered two LTL properties from the literature discussed with domain experts
- We simulated an incremental model design with TOrPEDO

# Usefulness (RQ4): Results

**Table 9.** LTL formulas checked on the 64 states (P)KS representing a sub-network of MAPK pathway.

| Property | Initial Model | Revision 1 | Revision 2 | Revision 3 | Revision 4 |
|---|---|---|---|---|---|
| $\phi_1$ | ANALYSIS $= \top$ | RE-CHECK $= \top$ | RE-CHECK $= \top$ | ANALYSIS $= \top$ | ANALYSIS $= \top$ |
| $\phi_2$ | ANALYSIS $= ?$ | RE-CHECK $= ?$ | RE-CHECK $= ?$ | ANALYSIS $= \top$ | ANALYSIS $= \top$ |
| $\phi_3$ | ANALYSIS $= \bot$ | RE-CHECK $= \bot$ | RE-CHECK $= \bot$ | ANALYSIS $= \bot$ | ANALYSIS $= \top$ |

# Usefulness (RQ4): Results

**Table 9.** LTL formulas checked on the 64 states (P)KS representing a sub-network of MAPK pathway.

| Property | Initial Model | Revision 1 | Revision 2 | Revision 3 | Revision 4 |
|---|---|---|---|---|---|
| $\phi_1$ | ANALYSIS = $\top$ | RE-CHECK = $\top$ | RE-CHECK = $\top$ | ANALYSIS = $\top$ | ANALYSIS = $\top$ |
| $\phi_2$ | ANALYSIS = ? | RE-CHECK = ? | RE-CHECK = ? | ANALYSIS = $\top$ | ANALYSIS = $\top$ |
| $\phi_3$ | ANALYSIS = $\bot$ | RE-CHECK = $\bot$ | RE-CHECK = $\bot$ | ANALYSIS = $\bot$ | ANALYSIS = $\top$ |

- We evaluated three properties on five models

# Usefulness (RQ4): Results

**Table 9.** LTL formulas checked on the 64 states (P)KS representing a sub-network of MAPK pathway.

| Property | Initial Model | Revision 1 | Revision 2 | Revision 3 | Revision 4 |
|---|---|---|---|---|---|
| $\phi_1$ | ANALYSIS = $\top$ | RE-CHECK = $\top$ | RE-CHECK = $\top$ | ANALYSIS = $\top$ | ANALYSIS = $\top$ |
| $\phi_2$ | ANALYSIS = ? | RE-CHECK = ? | RE-CHECK = ? | ANALYSIS = $\top$ | ANALYSIS = $\top$ |
| $\phi_3$ | ANALYSIS = $\bot$ | RE-CHECK = $\bot$ | RE-CHECK = $\bot$ | ANALYSIS = $\bot$ | ANALYSIS = $\top$ |

- We evaluated three properties on five models
- We run the analysis three times

# Usefulness (RQ4): Results

**Table 9.** LTL formulas checked on the 64 states (P)KS representing a sub-network of MAPK pathway.

| Property | Initial Model | Revision 1 | Revision 2 | Revision 3 | Revision 4 |
|---|---|---|---|---|---|
| $\phi_1$ | ANALYSIS = $\top$ | RE-CHECK = $\top$ | RE-CHECK = $\top$ | ANALYSIS = $\top$ | ANALYSIS = $\top$ |
| $\phi_2$ | ANALYSIS = ? | RE-CHECK = ? | RE-CHECK = ? | ANALYSIS = $\top$ | ANALYSIS = $\top$ |
| $\phi_3$ | ANALYSIS = $\perp$ | RE-CHECK = $\perp$ | RE-CHECK = $\perp$ | ANALYSIS = $\perp$ | ANALYSIS = $\top$ |

- We evaluated three properties on five models
- We run the analysis three times and used the syntactic check twice

# Usefulness (RQ4): Results

**Table 9.** LTL formulas checked on the 64 states (P)KS representing a sub-network of MAPK pathway.

| Property | Initial Model | Revision 1 | Revision 2 | Revision 3 | Revision 4 |
|---|---|---|---|---|---|
| $\phi_1$ | ANALYSIS = $\top$ | RE-CHECK = $\top$ | RE-CHECK = $\top$ | ANALYSIS = $\top$ | ANALYSIS = $\top$ |
| $\phi_2$ | ANALYSIS = ? | RE-CHECK = ? | RE-CHECK = ? | ANALYSIS = $\top$ | ANALYSIS = $\top$ |
| $\phi_3$ | ANALYSIS = $\bot$ | RE-CHECK = $\bot$ | RE-CHECK = $\bot$ | ANALYSIS = $\bot$ | ANALYSIS = $\top$ |

- We evaluated three properties on five models
- We run the analysis three times and used the syntactic check twice
- The topological proofs provide useful information

# Usefulness (RQ4): Results

The answer to **RQ4** is that the topological proofs and counterexamples provided by TOrPEDO effectively supported the development of a (P)KS representing a gene regulatory network.

2021

# TOrPEDO: Witnessing Model Correctness with Topological Proofs

Formal Aspects of Computing (FAOC)


Menghi, Claudio

Rizzi, Alessandro Maria

Bernasconi, Anna

Spoletini, Paola


# Reflections

# Correctness

The algorithm is correct if the LTL clauses are contradicting

- The correctness depends on the value of k
- If k is higher than the completeness threshold, the LTL clauses are contradicting

* Linear encodings of bounded LTL model checking
Schuppan V, Latvala T, Junttila T, Heljanko K, Biere A (2006)
Log Methods Comput Sci, 2, Episciences.org

* Linear completeness thresholds for bounded model checking.
Kroening D, Ouaknine J, Strichman O, Wahl T, Worrell J (2011)
Computer aided verification, Springer

* Completeness and complexity of bounded model checking.
Clarke E, Kroening D, Ouaknine J, Strichman O (2004)
International conference on verification, model checking, and abstract interpretation, Springer

## Practical Guidelines

- Designers can
  - initially choose a value for k that is reasonably large
  - increase or decrease the value of k depending on
    - the efficiency of the analysis
    - the importance of the soundness

# Why Faster

- TORPEDO-MUP is FPSPACE complete, TORPEDO-SMT is NP-complete

# Why Faster

- The Z3 Theorem Prover offers a mature technology;
    - an industry-strength tool,
    - awarded by ETAPS (Test of Time Award) and ACM SIGPLAN (Programming Languages Software Award)

# Trace Diagnostics for Signal-based Temporal Properties

IEEE Transactions on Software Engineering (TSE)

2023

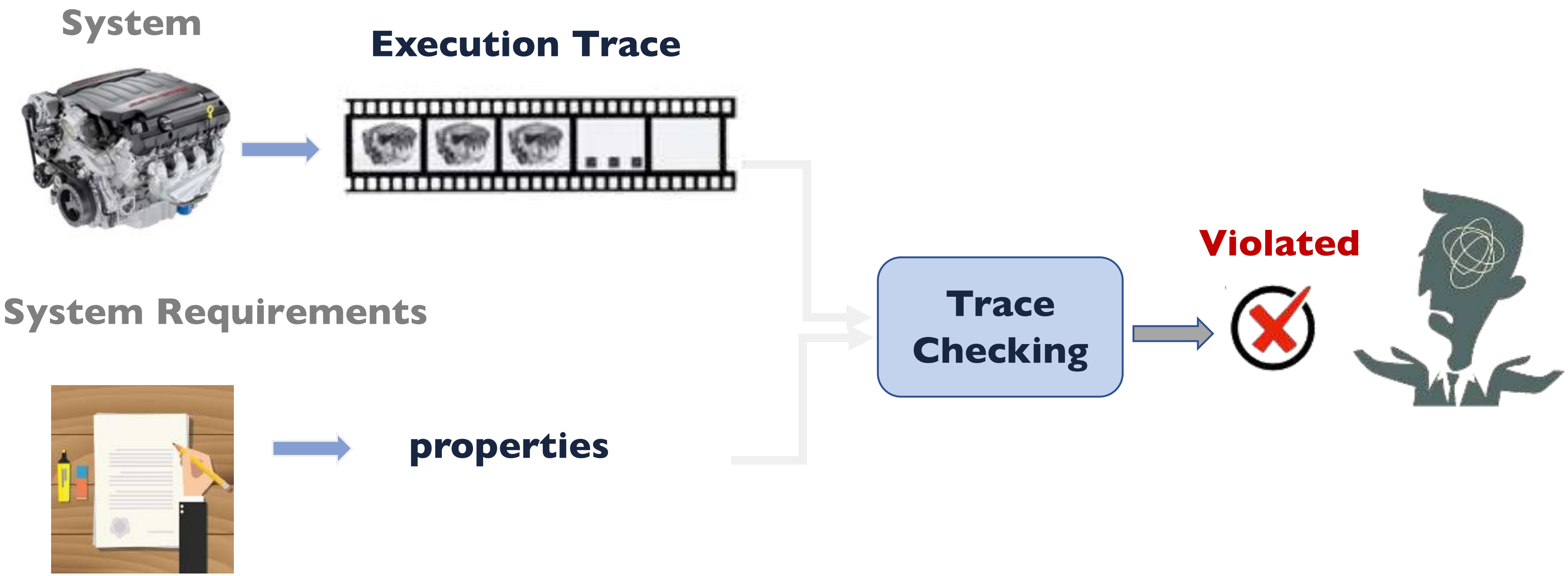
Boufaied, Chaima

Menghi, Claudio

Bianculli, Domenico

Briand, Lionel C

# Trace Diagnostics



# Problem

How do we explain why
a property is violated by a trace?

# Contribution (TD-SB-TemPsy)

TD-SB-TemPsy: A trace-diagnostic approach for signal-based temporal properties.

- analyzes a trace and a property violated by the trace;
- provides an explanation for the property violation.

# Contribution (TD-SB-TemPsy)

TD-SB-TemPsy relies on

- violation causes and
- diagnoses.

## Contribution (TD-SB-TemPsy)

Violation cause: characterizes one of the possible behaviors of the system that may lead to the property violation.

Diagnoses: information associated with the property violation

# Contribution (TD-SB-TemPsy)

Violation cause: characterizes one of the possible behaviors of the system that may lead to the property violation.

## Contribution (TD-SB-TemPsy)

Violation cause: characterizes one of the possible behaviors of the system that may lead to the property violation.

A violation cause should satisfy the following relation:

- if the violation cause holds, then the corresponding requirement should be violated

# Topological Proofs and Violation Clauses: Parallelism

A topological proof is a slice of the model that witnesses property satisfaction

A violation cause is a construct that if satisfied by a (slice) of the trace witnesses property violation

# Contribution (TD-SB-TemPsy)

The paper describes

- TD-SB-TemPsy, a trace-diagnostic approach for signal-based temporal properties expressed in SB-TemPsy-DSL,
- a methodology for defining violation causes and diagnoses, with formal guarantees of the soundness of the violation causes

# Contribution (TD-SB-TemPsy)

The paper describes

- a catalog of 34 violation causes, each associated with one diagnosis,
- evaluates TD-SBTemPsy on two datasets, including one industrial case study.

# TD-SB-TemPsy Evaluation

Evaluated with an Industrial Case Study

- 361 traces given by our industrial partner
- 98 requirements specified in SB-TemPsy-DSL
- Total: 35378 trace - property combinations

# TD-SB-TemPsy Evaluation

TD-SB-TemPsy yielded a diagnosis within a timeout of 1 minute for 83.66% of the combinations

# Reflections and Lessons Learned and Speculations

**2017**

**From Model Checking to a Temporal Proof for Partial Models**
International Conference on Software Engineering and Formal Methods (SEFM)
Bernasconi, Anna; **Menghi, Claudio**; Spoletini, Paola; Zuck, Lenore D; Ghezzi, Carlo

**2020**

**Integrating Topological Proofs with Model Checking to Instrument Iterative Design**
Fundamental Approaches to Software Engineering (FASE)
**Menghi, Claudio**; Rizzi, Alessandro Maria; Bernasconi, Anna

**2021**

**TOrPEDO: Witnessing Model Correctness with Topological Proofs**
Formal Aspects of Computing (FAOC)
**Menghi, Claudio**; Rizzi, Alessandro Maria; Bernasconi, Anna; Spoletini, Paola

**2023**

**Trace Diagnostics for Signal-based Temporal Properties**
IEEE Transactions on Software Engineering (TSE),
Boufaied, Chaima; **Menghi, Claudio**; Bianculli, Domenico; Briand, Lionel C

# Reflections and Lessons Learned and Speculations

## Reflection 1: There is a synergy between theory and practice

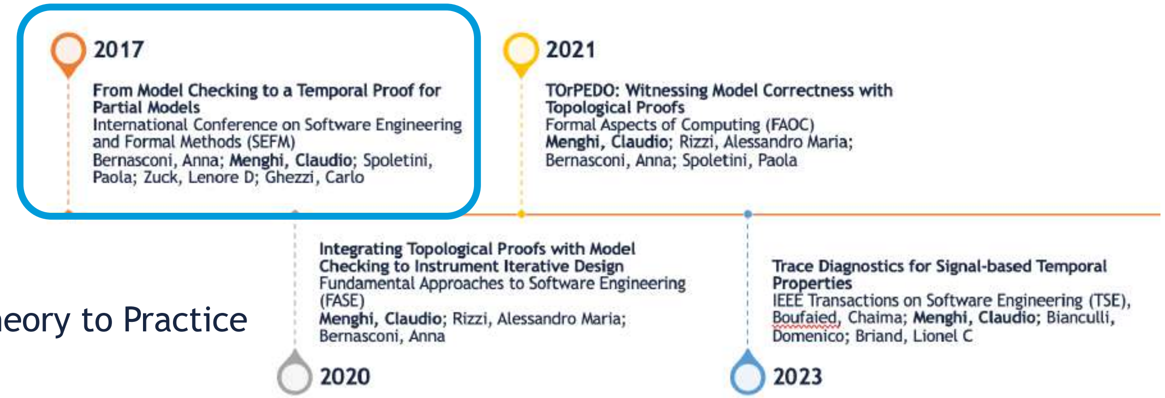

Automated Verification of Cyber-Physical Systems: From Theory to Practice
Workshop on Software Reliability for Madrid Flight on Chip
https://flightonchip.es/workshop19/

Verification and Validation: from Theory to Practice and Back Again
November 6th, 2020
https://www.deib.polimi.it/eng/events/details/2111

# Reflections and Lessons Learned and Speculations

## Reflection 1: There is a synergy between theory and practice

No Implementation




Automated Verification of Cyber-Physical Systems: From Theory to Practice
Workshop on Software Reliability for Madrid Flight on Chip
https://flightonchip.es/workshop19/

Verification and Validation: from Theory to Practice and Back Again
November 6th, 2020
https://www.deib.polimi.it/eng/events/details/2111

# Reflections and Lessons Learned and Speculations

## **Reflection 1**: There is a synergy between theory and practice


Limitations on the efficiency

Automated Verification of Cyber-Physical Systems: From Theory to Practice
Workshop on Software Reliability for Madrid Flight on Chip
https://flightonchip.es/workshop19/

Verification and Validation: from Theory to Practice and Back Again
November 6th, 2020
https://www.deib.polimi.it/eng/events/details/2111

# Reflections and Lessons Learned and Speculations

**Reflection 1**: There is a synergy between theory and practice

Improvement of the efficiency


Automated Verification of Cyber-Physical Systems: From Theory to Practice
Workshop on Software Reliability for Madrid Flight on Chip
https://flightonchip.es/workshop19/

Verification and Validation: from Theory to Practice and Back Again
November 6th, 2020
https://www.deib.polimi.it/eng/events/details/2111

# Reflections and Lessons Learned and Speculations

## Reflection 1: There is a synergy between theory and practice


Evaluation on the industrial domain

Automated Verification of Cyber-Physical Systems: From Theory to Practice
Workshop on Software Reliability for Madrid Flight on Chip
https://flightonchip.es/workshop19/

Verification and Validation: from Theory to Practice and Back Again
November 6th, 2020
https://www.deib.polimi.it/eng/events/details/2111

# Reflections and Lessons Learned and Speculations

## Reflection 2: The results are teamwork

Bernasconi, Anna

Rizzi, Alessandro Maria

Bianculli, Domenico

Boufaied, Chaima

Spoletini, Paola

Ghezzi, Carlo

Zuck, Lenore D

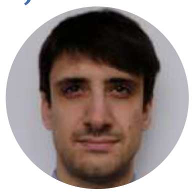
Briand, Lionel C

# Reflections and Lessons Learned and Speculations

## Reflection 2: The results are teamwork

Bernasconi, Anna

Rizzi, Alessandro Maria




They are first authors!

## Reflections and Lessons Learned and Speculations

**Reflection 3**: Some of the reviewers significantly helped us in improving the papers.

VMCAI 2019: REVIEW 3 (Reject)

*For LTL formulae, the separated normal form […] One can create an equisatisfiable normalized formula, but not an equivalent one. Why this should still work and how the reasons/understanding is explained using a non-equivalent formula is not discussed at all.*

It was indeed equivalent.
Thanks a lot!

## Reflections and Lessons Learned and Speculations

**Reflection 4**: Did we reach ``The Independence Day of Witnessing the Correctness of Systems''?

## Reflections and Lessons Learned and Speculations

**Reflection 4**: Did we reach ``The Independence Day of Witnessing the Correctness of Systems''?

Well, no, I think there is a lot of work that still to be done.

# Reflections and Lessons Learned and Speculations

Variety of the

modeling formalisms

## Reflections and Lessons Learned and Speculations

Variety of the
modeling formalisms

Variety of the Requirements
Specification Languages

# Reflections and Lessons Learned and Speculations

Variety of the
modeling formalisms

Variety of the Requirements
Specification Languages

Trade-off
Expressiveness and
Performances

# Reflections and Lessons Learned and Speculations

Variety of the
modeling formalisms

Variety of the Requirements
Specification Languages

Trade-off
Expressiveness and
Performances

Developing
Techniques that are
Complete

## Reflections and Lessons Learned and Speculations

Variety of the modeling formalisms

Variety of the Requirements Specification Languages

Trade-off Expressiveness and Performances

Usability for the End Users

Developing Techniques that are Complete

# Reflections and Lessons Learned and Speculations

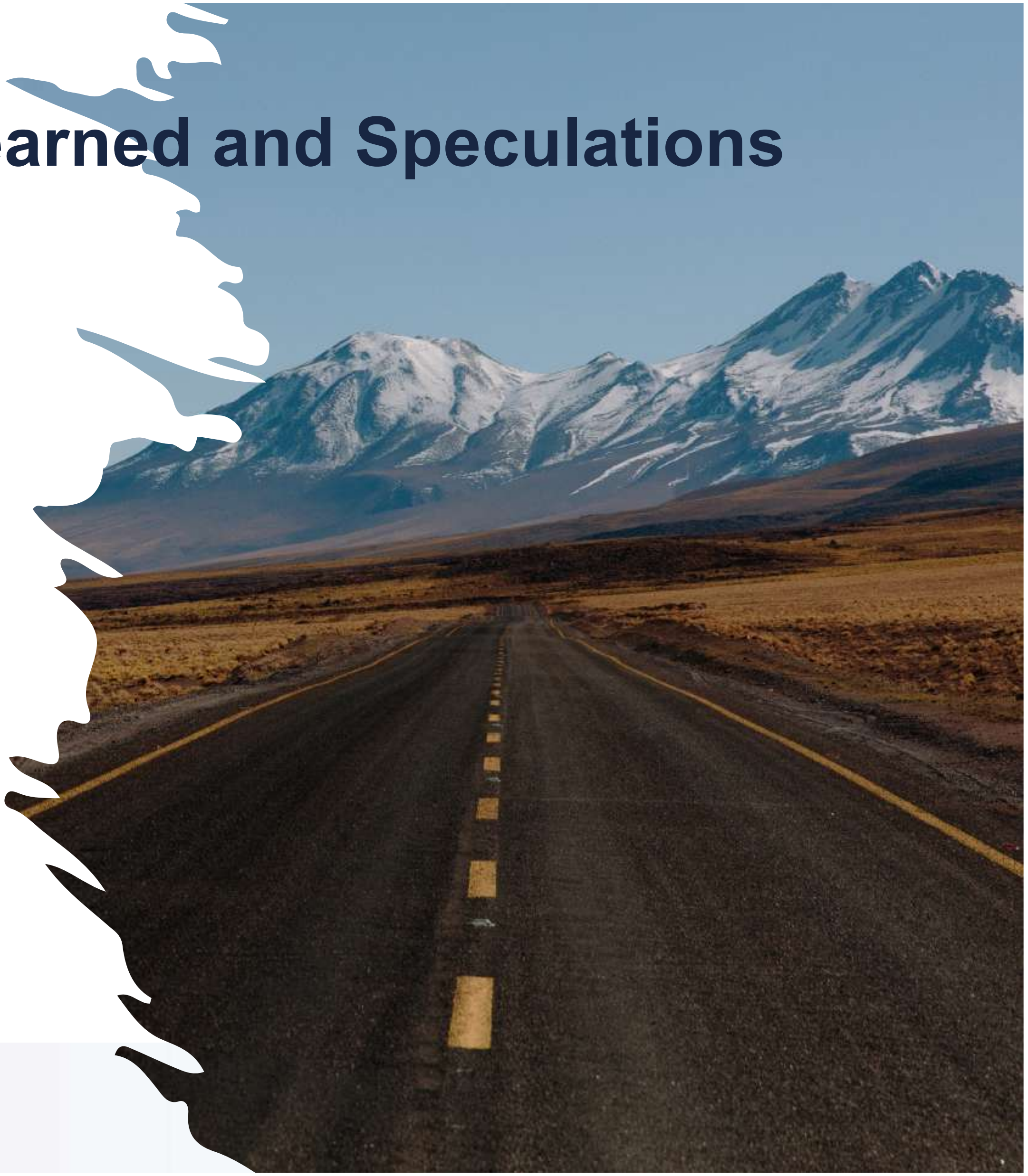
Reaching

````The Independence Day of Witnessing

the Correctness of Systems’’’’

is a journey, everyone is invited!

**Enjoy the trip!**


````

UNIVERSITÀ
DEGLI STUDI
DI BERGAMO

Dipartimento
di Ingegneria Gestionale,
dell'Informazione e della Produzione

# The Independence Day of Witnessing the Correctness of Systems: From Topological Proofs and Beyond

Speaker

Claudio MENGHI

BCS FACS (Formal Aspects of Computing Science)

Date: 4th July 2023