End-to-End Validation of Input Model Transformations in Model Checking Tools

Zsófia Ádám

Budapest University of Technology and Economics

MŰEGYETEM 1782

BME FACULTY OF ELECTRICAL ENGINEERING AND INFORMATICS

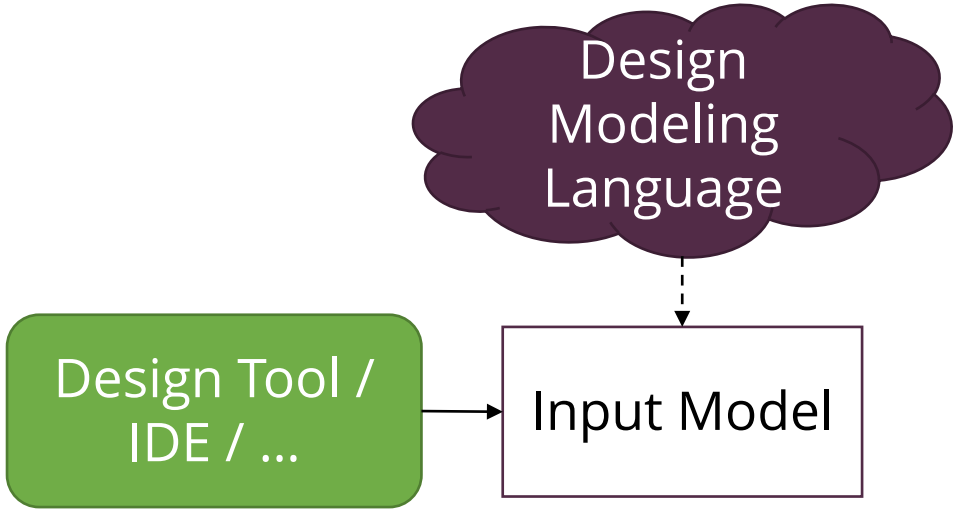
ftsrg

Critical Systems Research Group

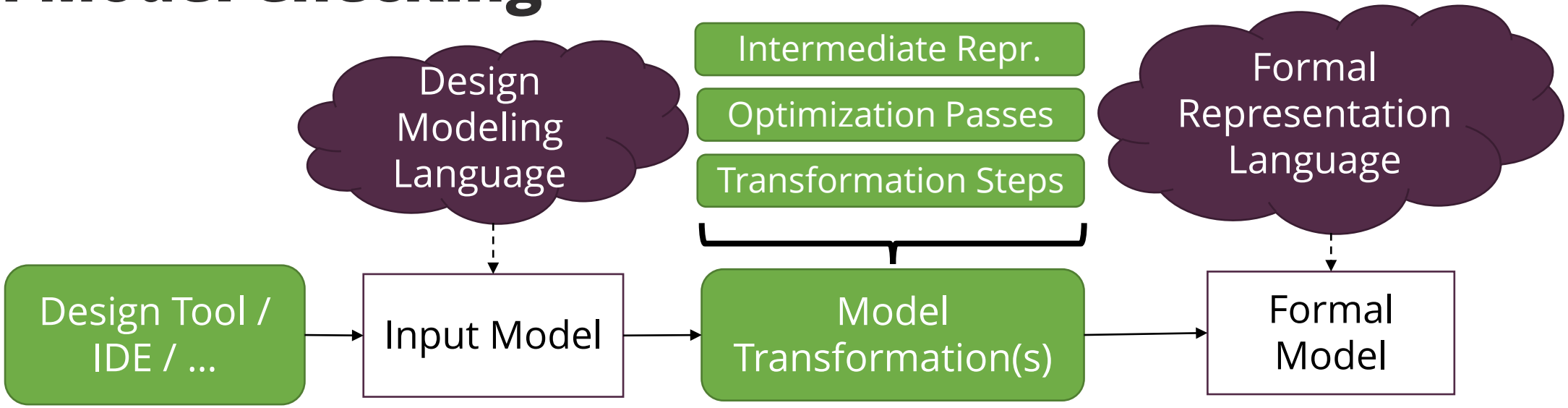
Motivation

Automatic Model Transformations in Model Checking

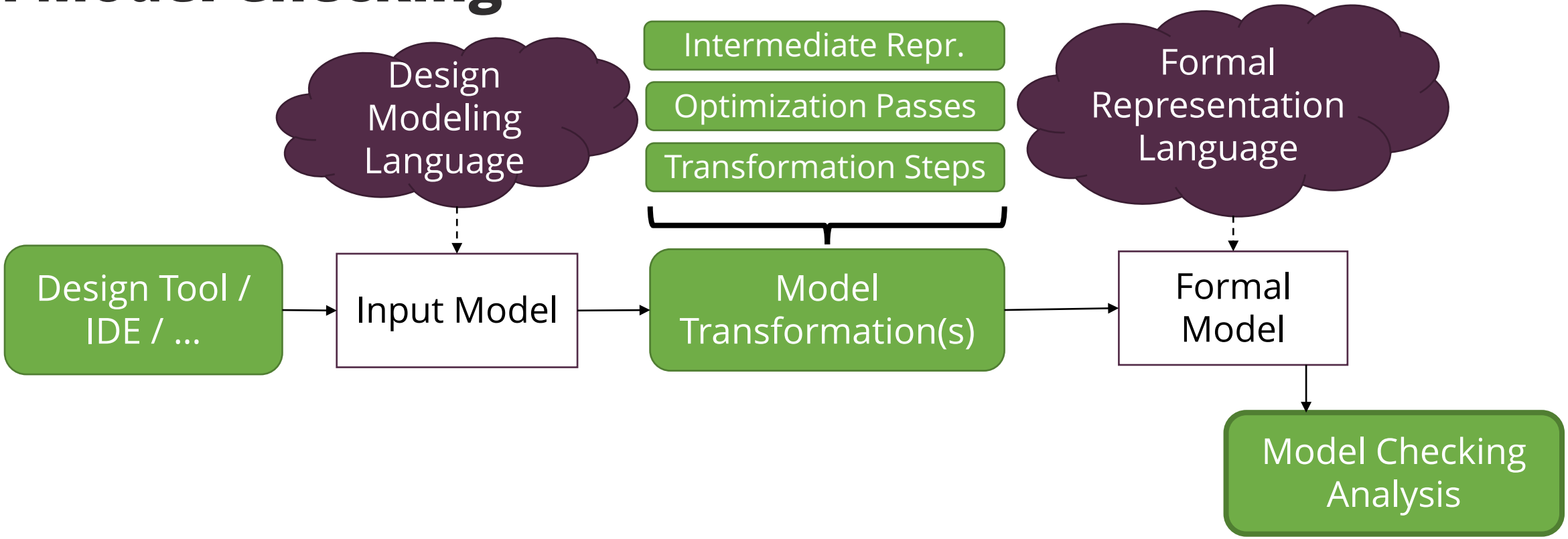
Automatic Model Transformations in Model Checking



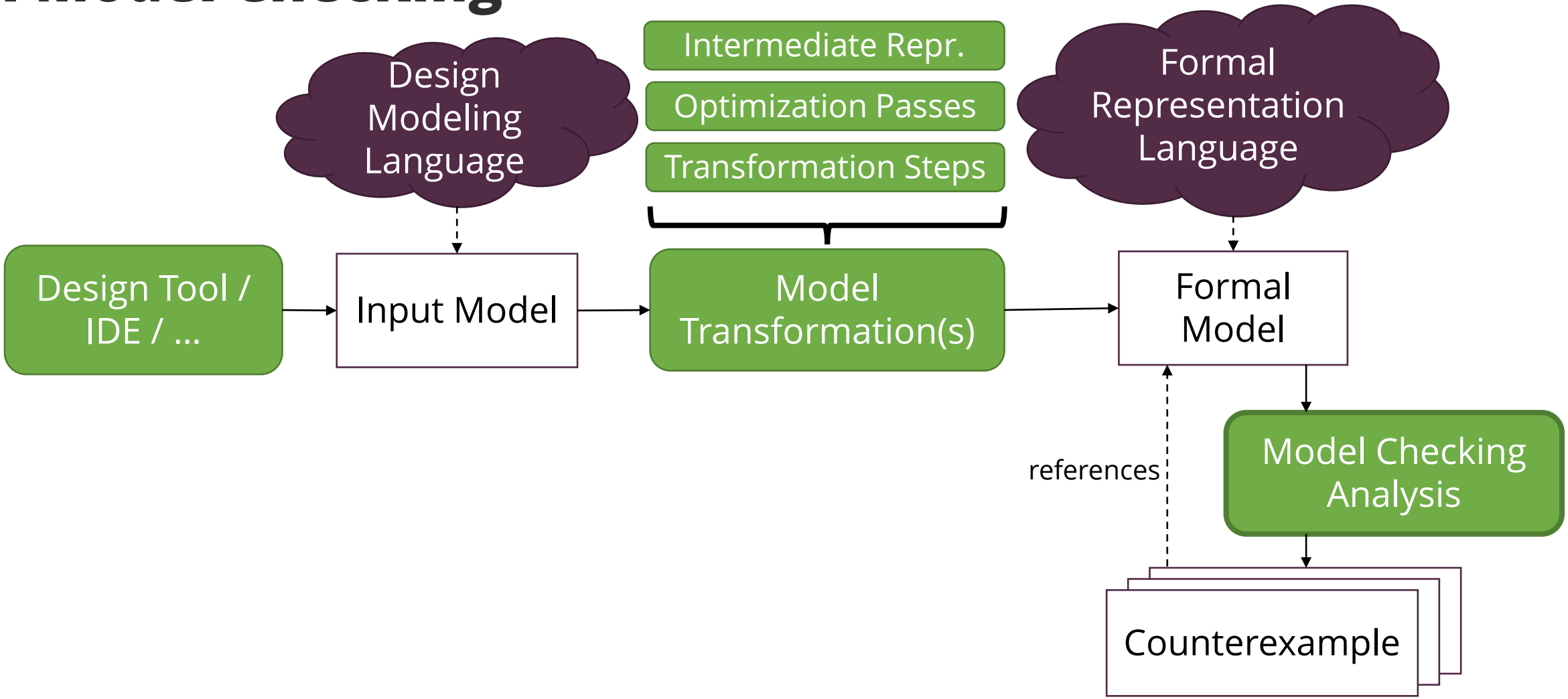
Automatic Model Transformations in Model Checking



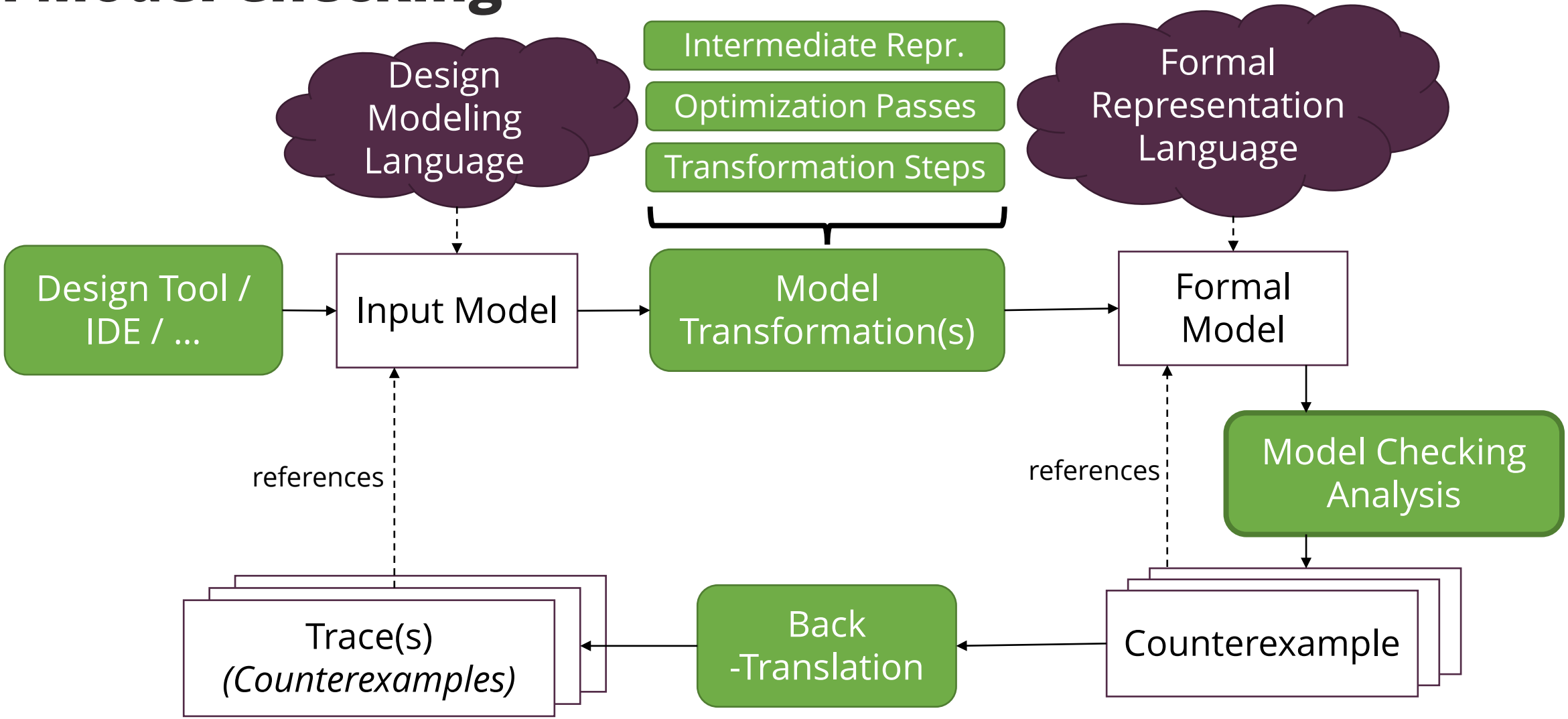
Automatic Model Transformations in Model Checking



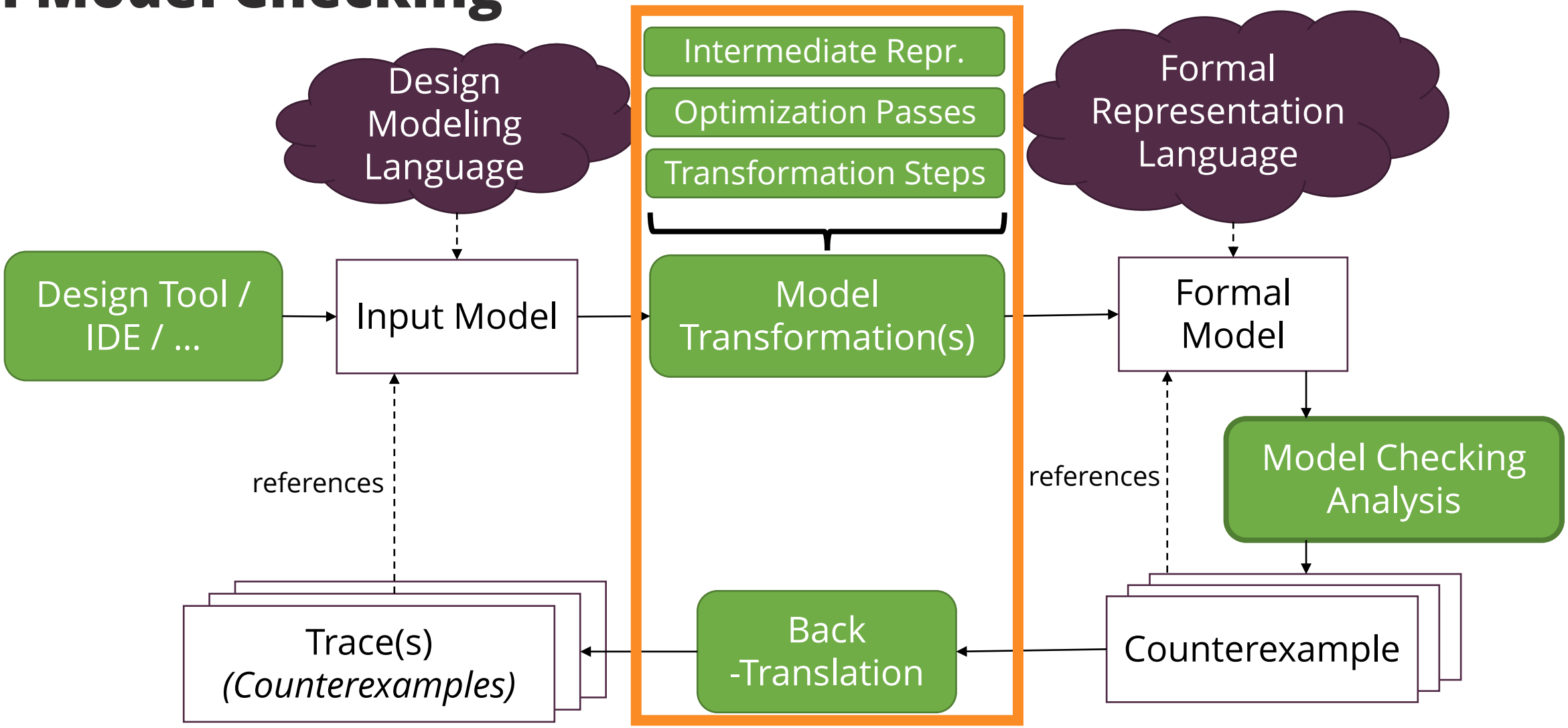
Automatic Model Transformations in Model Checking



Automatic Model Transformations in Model Checking



Automatic Model Transformations in Model Checking



Modeling Languages for Design and Verification

Modeling Languages for Design and Verification

Design Modeling Language

- Complex
- Lots of language elements, Syntactic sugar
- Human readable, Succinct
- Informal / Semi-formal Semantics

Modeling Languages for Design and Verification

Design Modeling Language

- Complex
- Lots of language elements, Syntactic sugar
- Human readable, Succinct
- Informal / Semi-formal Semantics

Formal representation

- Few language elements
- Contains all necessary information for model checker
- Verbose, not necessarily readable
- Formal Semantics

Modeling Languages for Design and Verification

Design Modeling Language

- Complex
- Lots of language elements, Syntactic sugar
- Human readable, Succinct
- Informal / Semi-formal Semantics

Formal representation

- Few language elements
- Contains all necessary information for model checker
- Verbose, not necessarily readable
- Formal Semantics

More and more model transformation chains!

Modeling Languages for Design and Verification

Such as…

- **C**, LLVM IR
- **Verilog**, **AIGER, Btor2**
- **UML, SysML**, **(XSTS)**
- **PLC code**, **(CFA)**
- Boogie
- **Promela**
- **SMV**
- Automata formats (*tool specific*)
  - **Timed Automata (UPPAAL)**
  - **CFA, STS (Theta)**
  - **GOTO (CBMC)**
- MoXI
- …

Design Modeling Language

- Complex
- Lots of language elements, Syntactic sugar
- Human readable, Succinct
- Informal / Semi-formal Semantics

Formal representation

- Few language elements
- Contains all necessary information for model checker
- Verbose, not necessarily readable
- Formal Semantics

More and more model transformation chains!

Example - Gamma Statechart Composition Framework

Example - Gamma Statechart Composition Framework

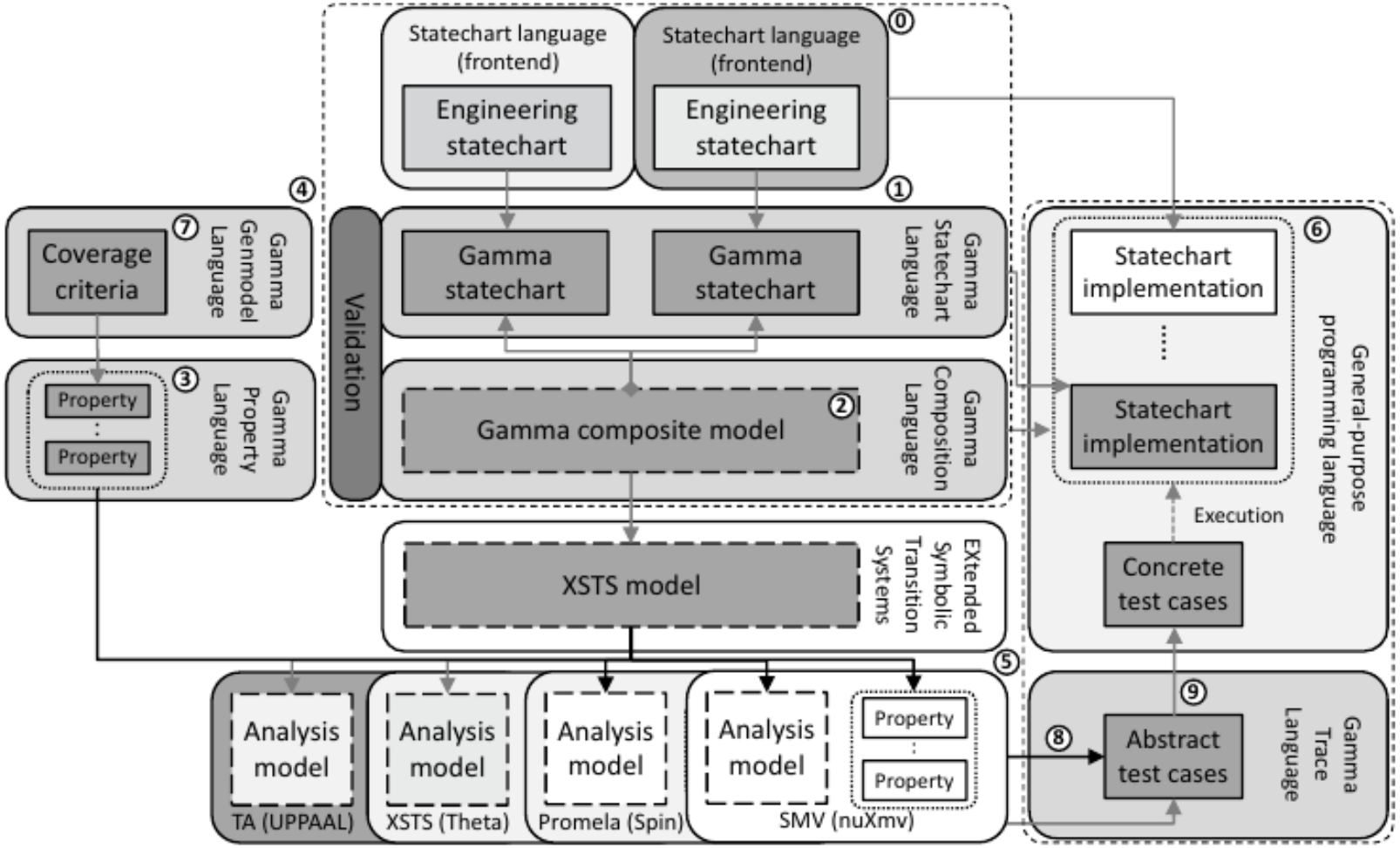


Figure 2: Modeling languages and model transformation chains of our MBT approach in the Gamma framework.

Example - Gamma Statechart Composition Framework

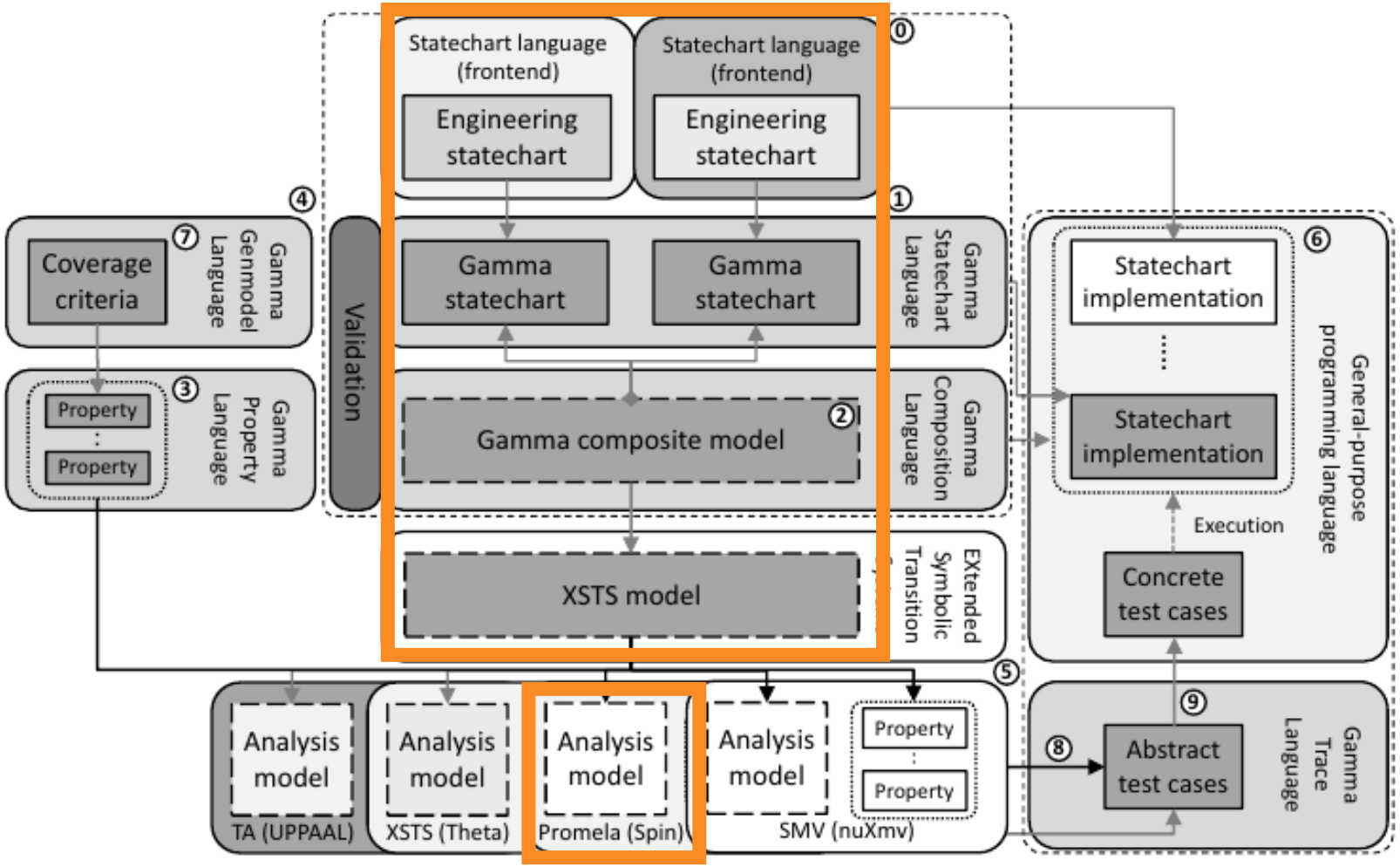


Figure 2: Modeling languages and model transformation chains of our MBT approach in the Gamma framework.

Example - Gamma Statechart Composition Framework

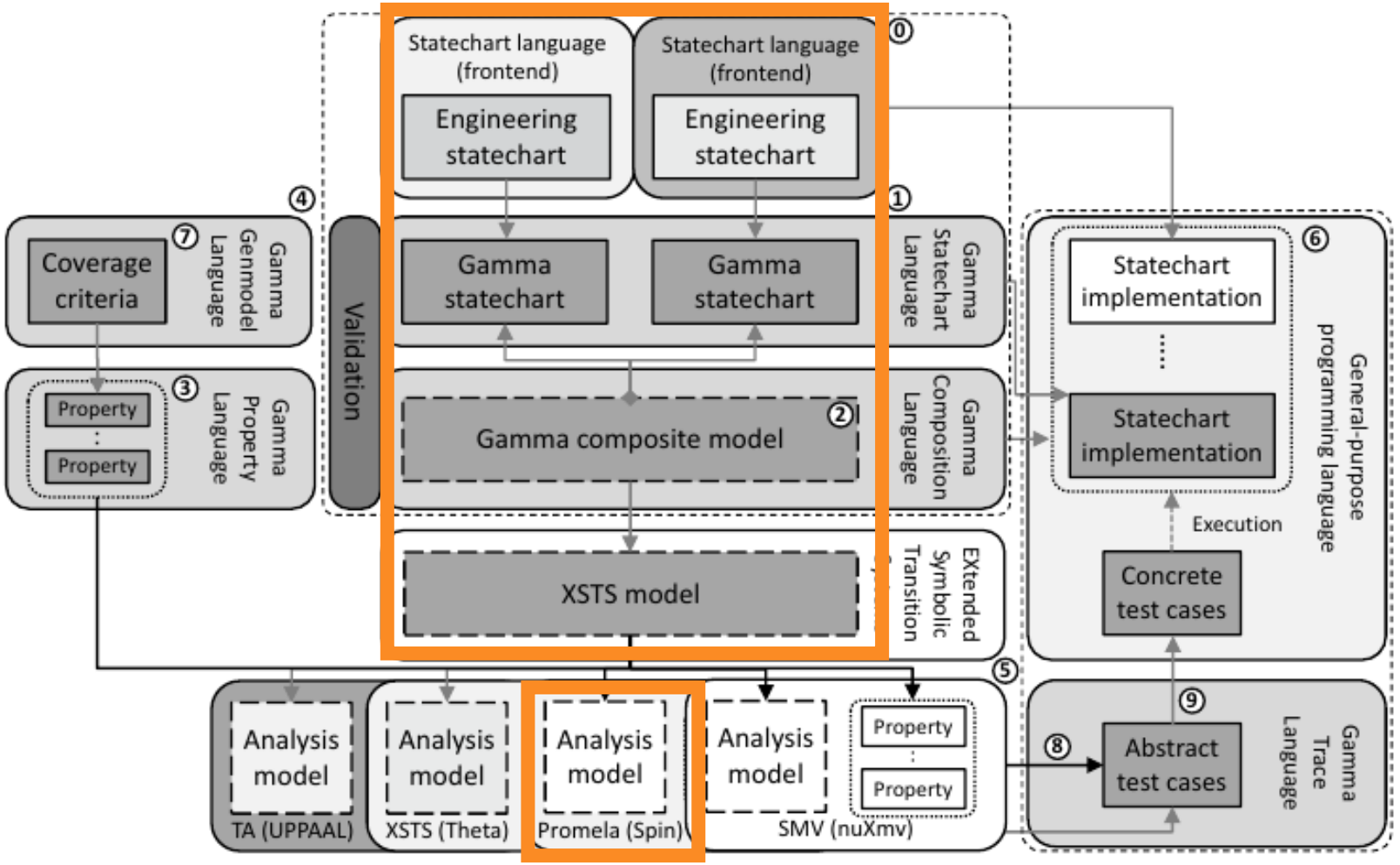
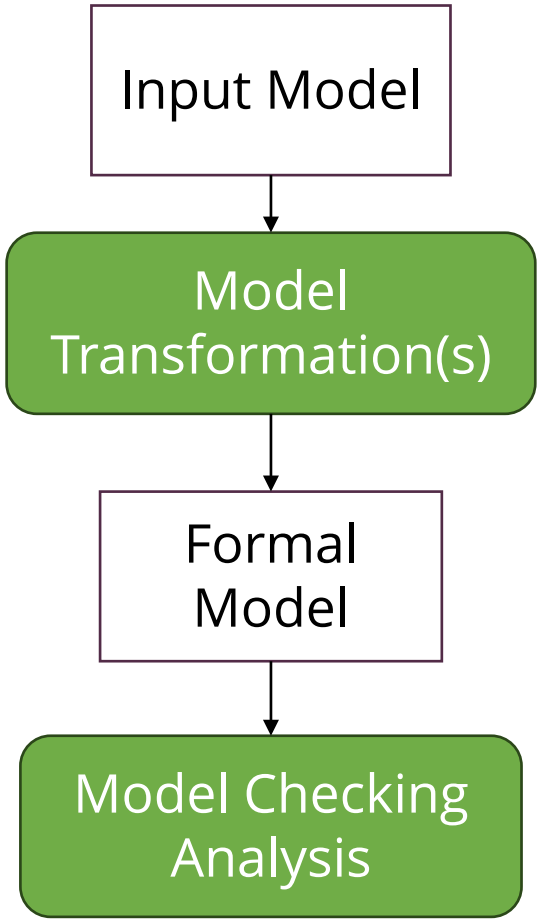


Figure 2: Modeling languages and model transformation chains of our MBT approach in the Gamma framework.

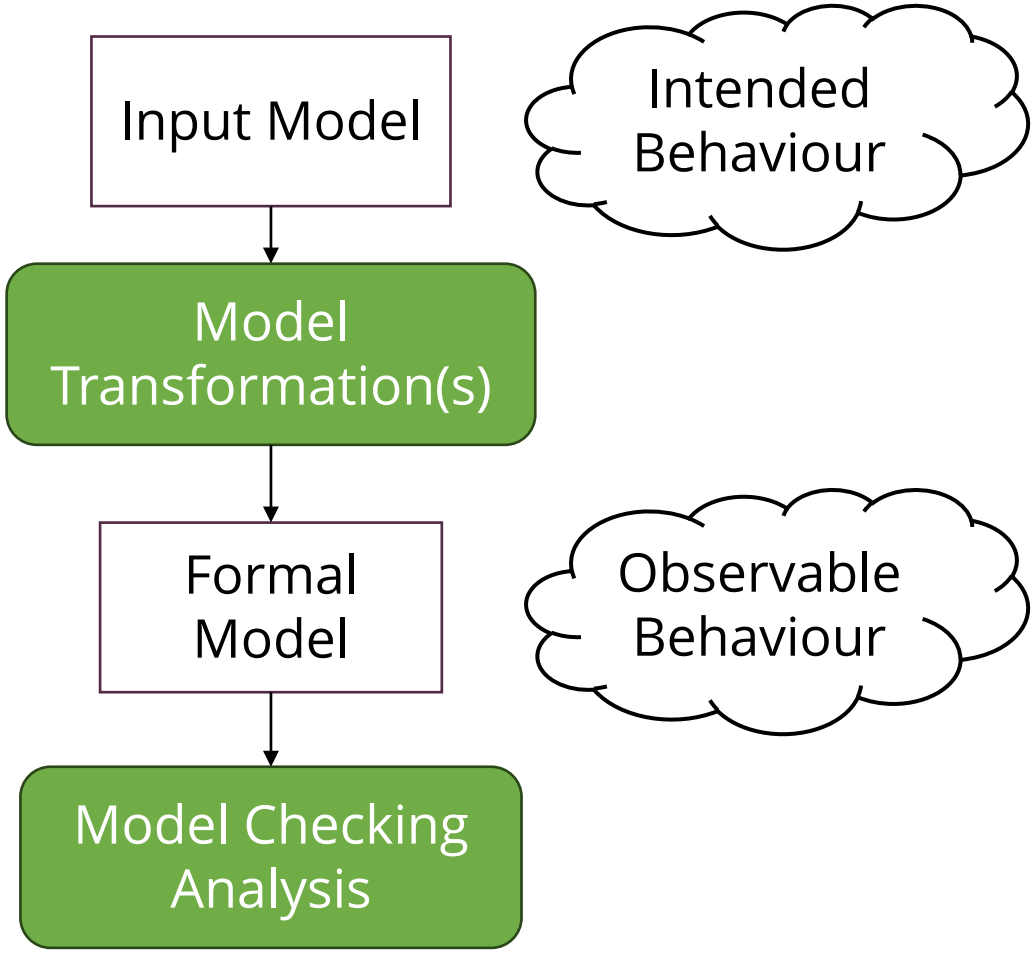
High Implementation Complexity:

- Viatra queries
- Xtend
- Ecore models

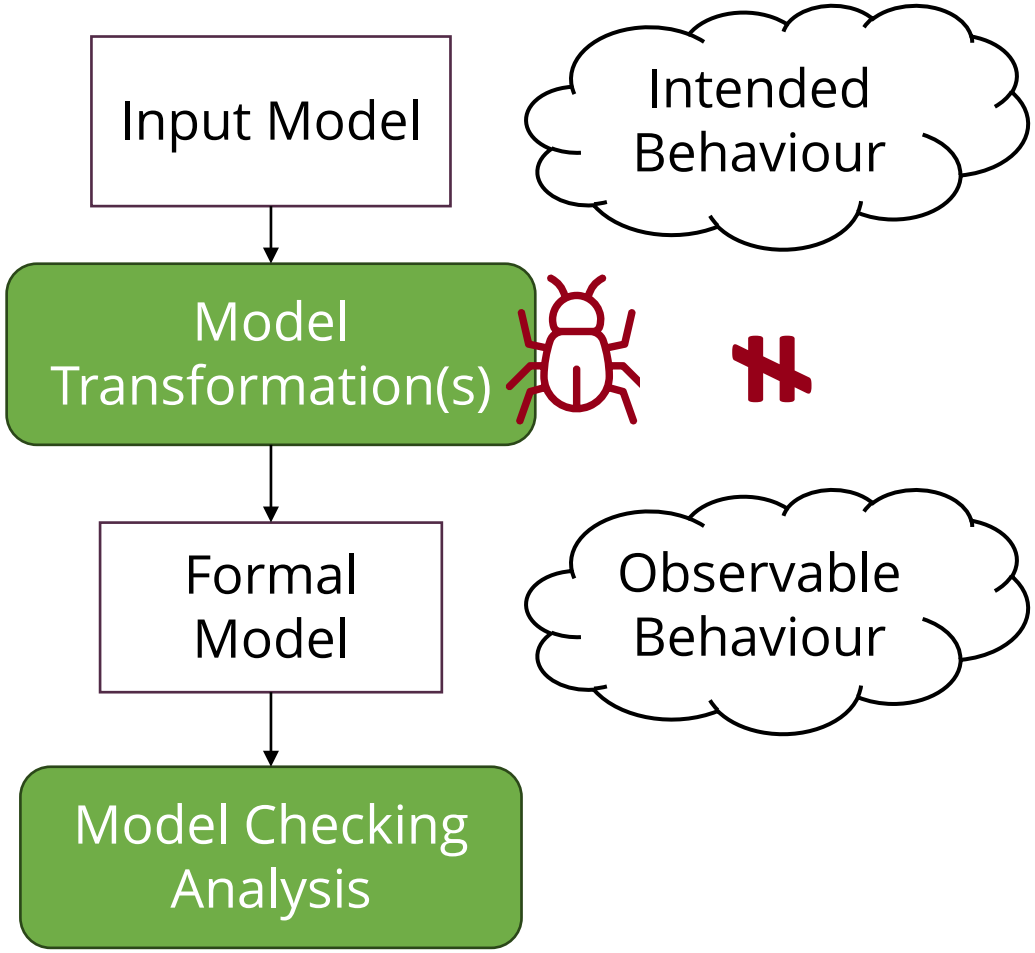
Fault Models [1] for Model Transformations



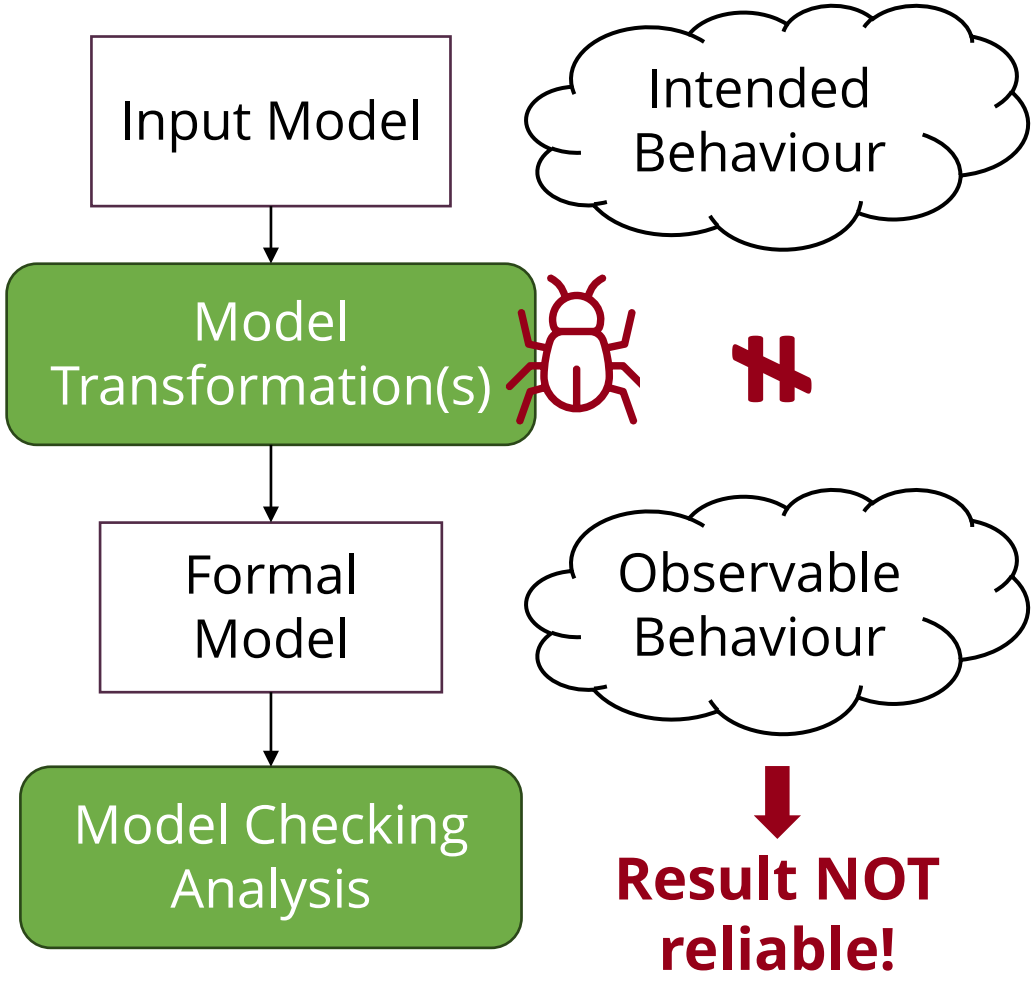
Fault Models [1] for Model Transformations



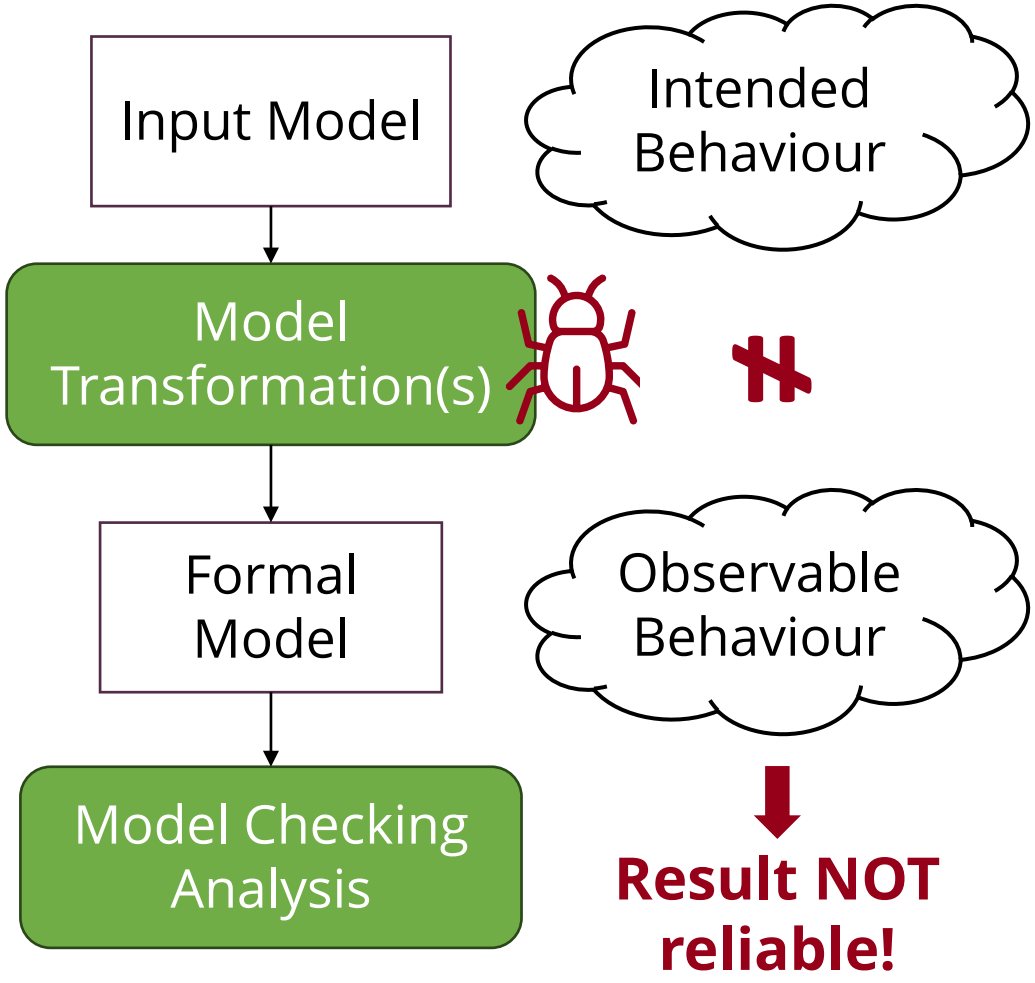
Fault Models [1] for Model Transformations



Fault Models [1] for Model Transformations

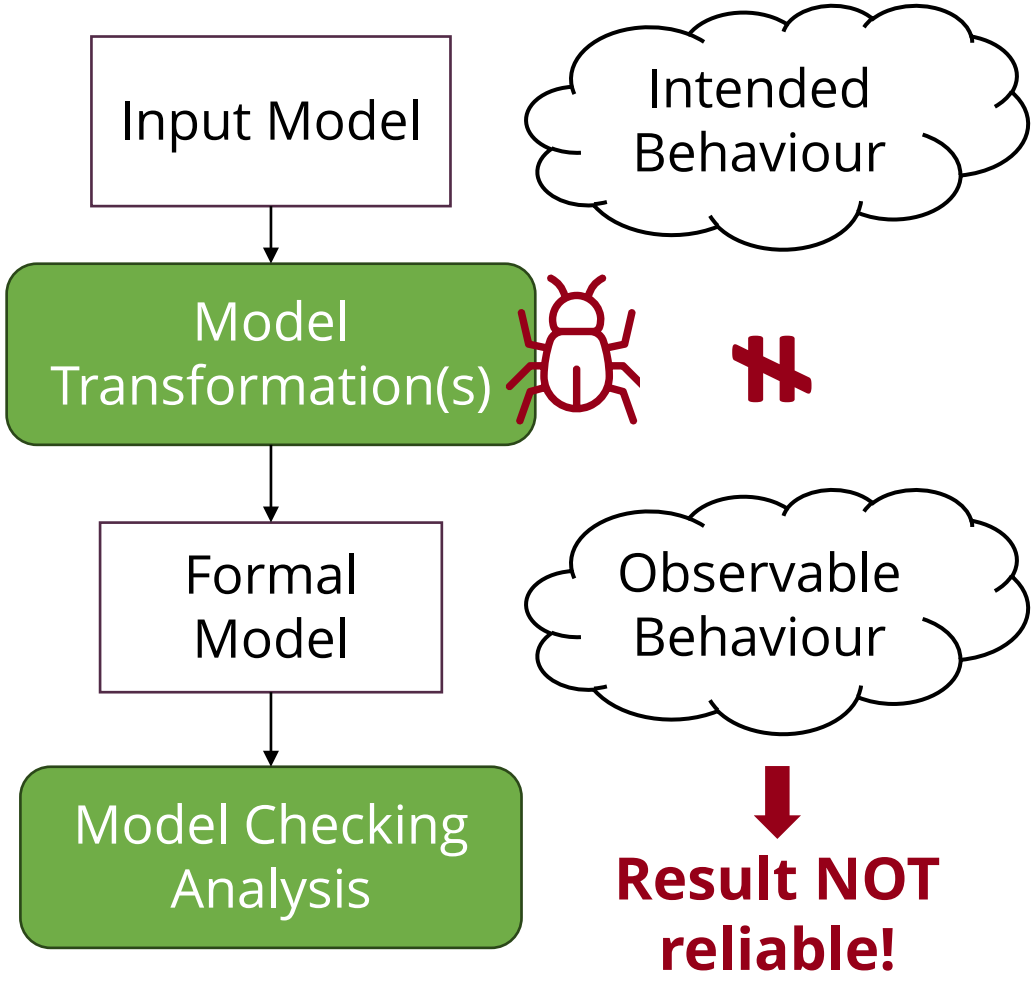


Fault Models [1] for Model Transformations



Examples

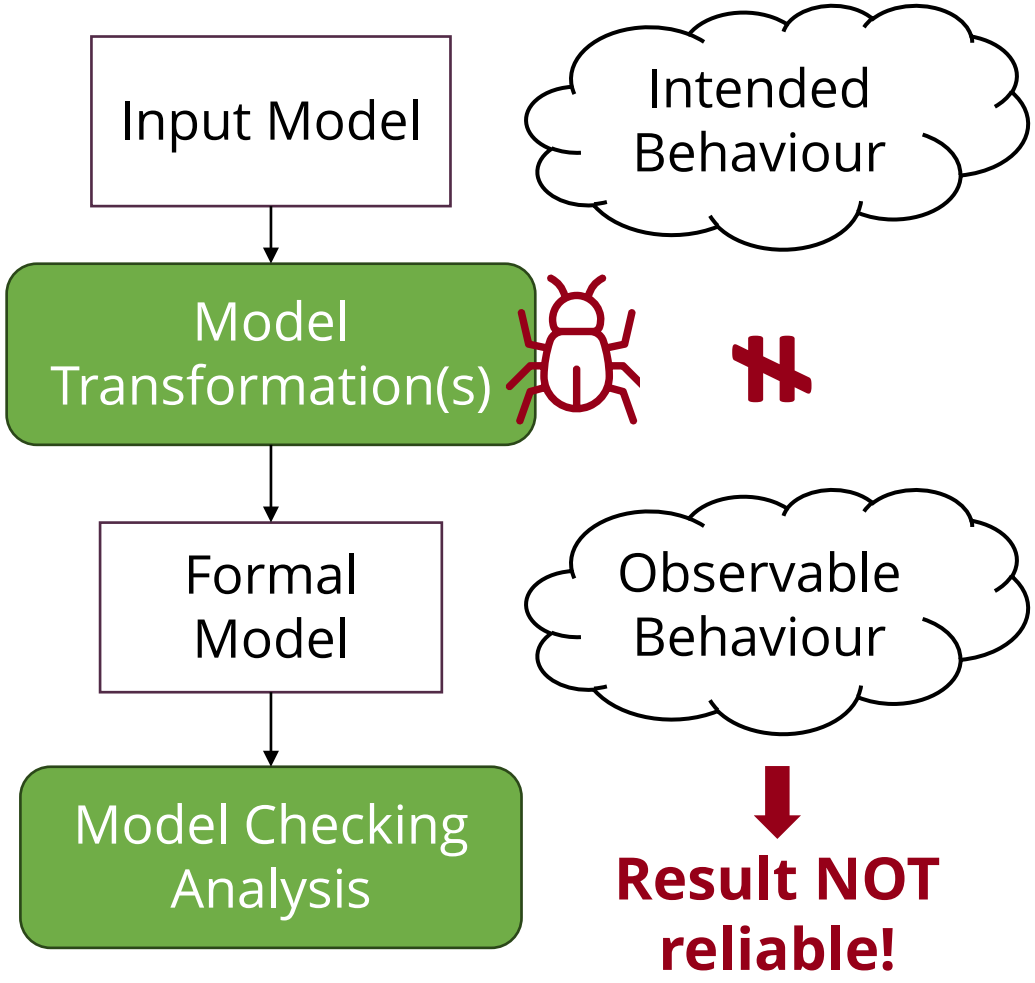
Fault Models [1] for Model Transformations



Examples

- Lost in Translation
  - *e.g., information lost due to SSA*

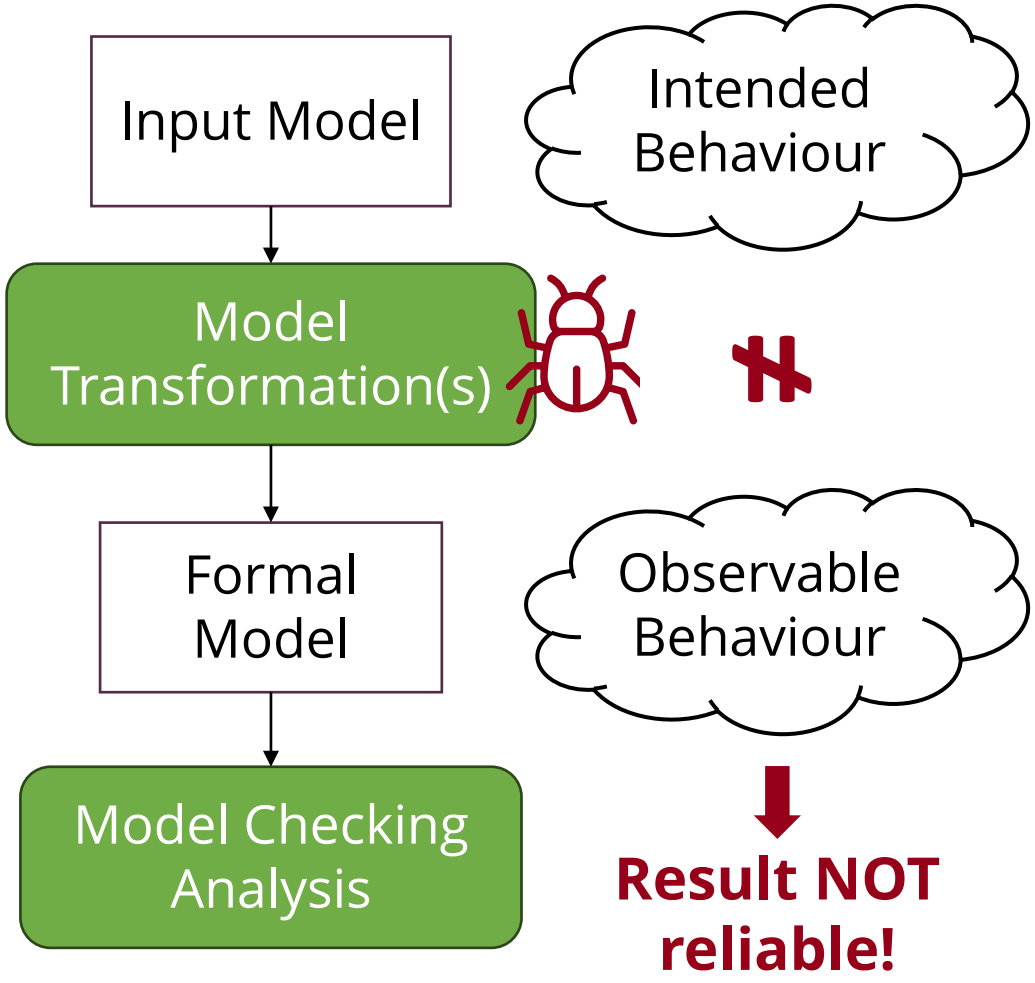
Fault Models [1] for Model Transformations



Examples

- Lost in Translation
  - *e.g., information lost due to SSA*
- Same Element, Different Semantics
  - *e.g., size of integers*

Fault Models [1] for Model Transformations



Examples

- Lost in Translation
  - *e.g., information lost due to SSA*
- Same Element, Different Semantics
  - *e.g., size of integers*
- Incorrect Translation of Element Combinations
  - *e.g., ordering of operators*

How to Find Model Transformation Issues?

How to Find Model Transformation Issues?

- **Example models**
  - Handwritten examples
  - Models from case studies
  - **Corner cases? Other language elements?**

How to Find Model Transformation Issues?

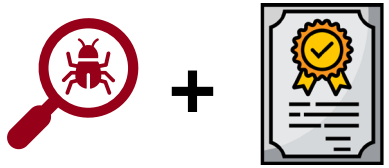
- **Example models**
  - Handwritten examples
  - Models from case studies
  - **Corner cases? Other language elements?**
- **Unit tests**
  - Usually present
  - **Complex toolchains - integration testing, end-to-end?**

How to Find Model Transformation Issues?

- **Example models**
  - Handwritten examples
  - Models from case studies
  - **Corner cases? Other language elements?**
- **Unit tests**
  - Usually present
  - **Complex toolchains - integration testing, end-to-end?**
- **Verify transformation rules** [2]
  - In practice: see Gamma example
    - **many layers and technologies**

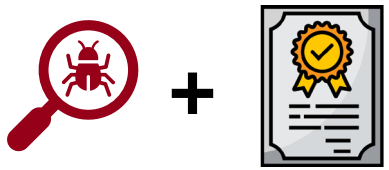
How to Find Model Transformation Issues?

- **Validate** (as in certify) results
  - e.g., SV-COMP software witnesses, btor2 violation witnesses
  - **Not always available**
    - No community format and tools
    - Some design models: not directly executable, validation harder
  - Also, why not **both**?



How to Find Model Transformation Issues?

- **Validate** (as in certify) results
  - e.g., SV-COMP software witnesses, btor2 violation witnesses
  - **Not always available**
    - No community format and tools
    - Some design models: not directly executable, validation harder
  - Also, why not **both**?



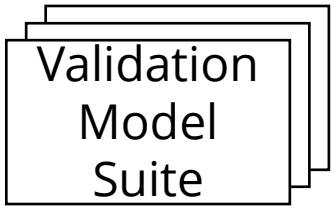
- **E2E validation**
  - **Validation Suites** *(compiler testing)* [3]
  - Idea from **testing with model checkers** [4]
    - Coverage criteria *(e.g., transition or decision coverage)*
    - Generate properties, check resulting counterexamples

Validation Workflow Proposals

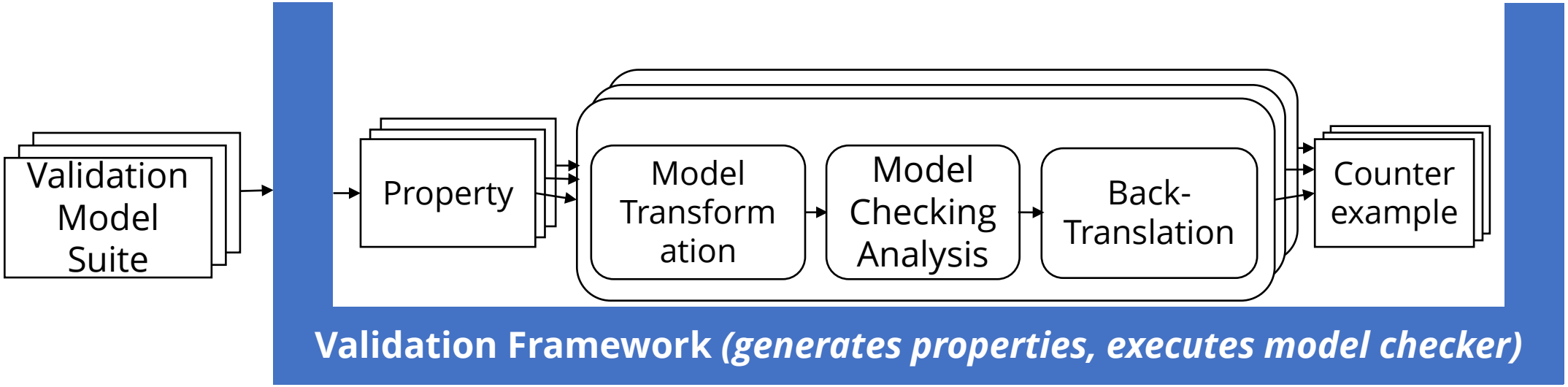
E2E Validation Method Proposal 1.

E2E Validation Method Proposal 1.

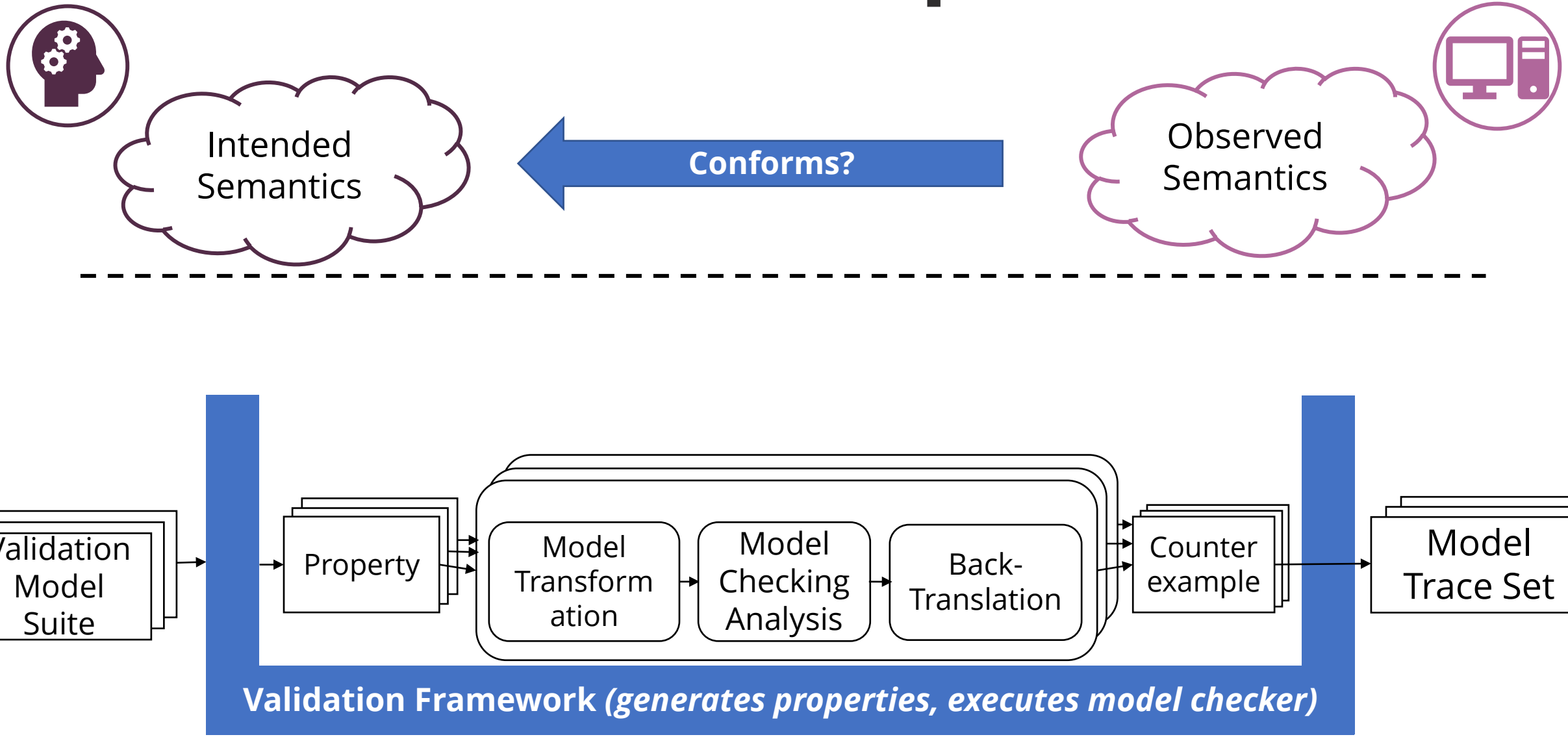
E2E Validation Method Proposal 1.



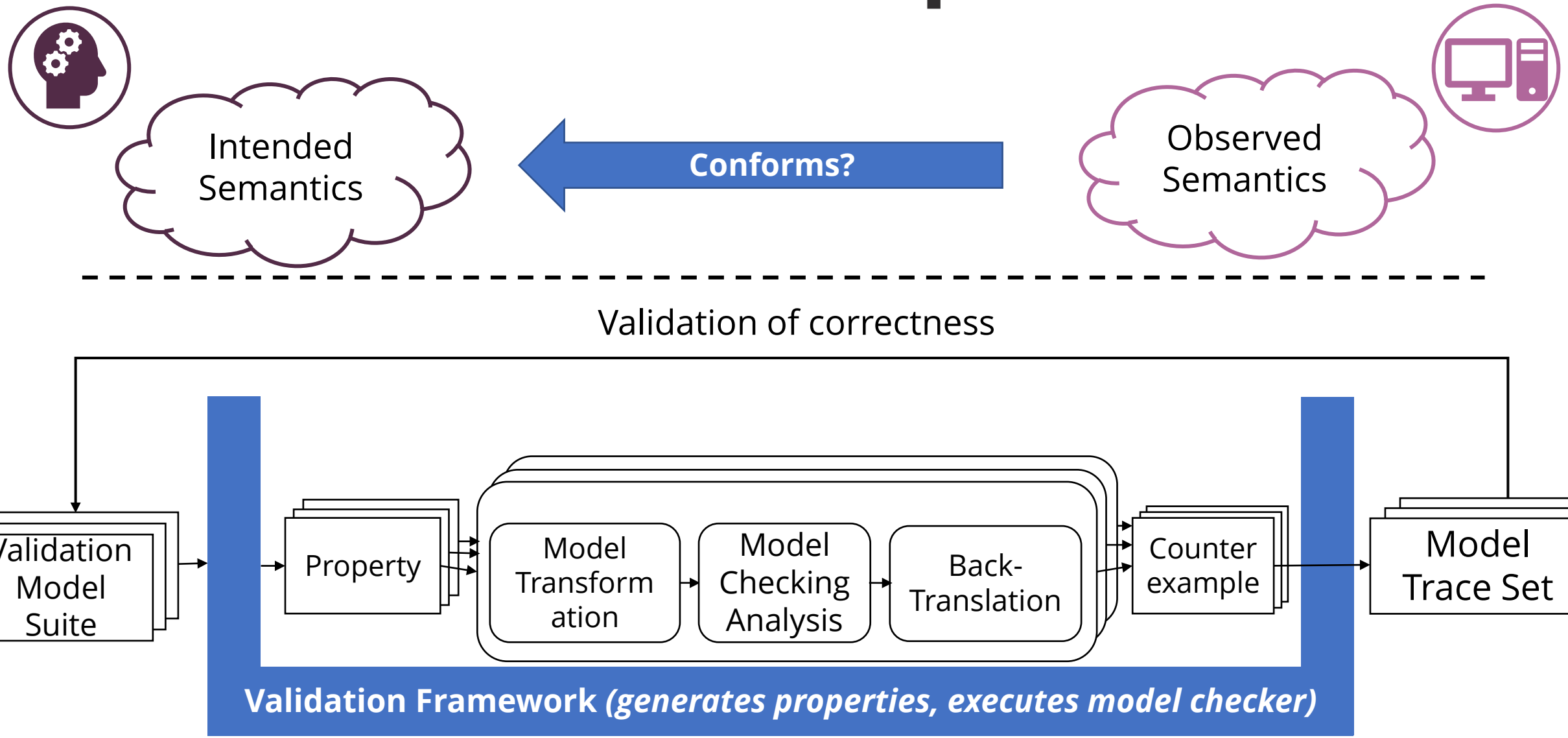
E2E Validation Method Proposal 1.



E2E Validation Method Proposal 1.



E2E Validation Method Proposal 1.



How to Create Properties?

How to Create Properties?

- Based on **coverage criteria** (Testing with Model Checking [4])

How to Create Properties?

- Based on **coverage criteria** (Testing with Model Checking [4])
  - State, transition, decision – even **path coverage** [5]

How to Create Properties?

- Based on **coverage criteria** (Testing with Model Checking [4])
  - State, transition, decision – even **path coverage** [5]
  - Can **generate** properties

How to Create Properties?

- Based on **coverage criteria** (Testing with Model Checking [4])
  - State, transition, decision – even **path coverage** [5]
  - Can **generate** properties
  - Can not necessarily show more **specific behaviour**

How to Create Properties?

- Based on **coverage criteria** (Testing with Model Checking [4])
  - State, transition, decision – even **path coverage** [5]
  - Can **generate** properties
  - Can not necessarily show more **specific behaviour**
    - *e.g., concurrency - which interleavings are possible*

How to Create Properties?

- Based on **coverage criteria** (Testing with Model Checking [4])
  - State, transition, decision – even **path coverage** [5]
  - Can **generate** properties
  - Can not necessarily show more **specific behaviour**
    - *e.g., concurrency - which interleavings are possible*
- Based on the **input model**

How to Create Properties?

- Based on **coverage criteria** (Testing with Model Checking [4])
  - State, transition, decision – even **path coverage** [5]
  - Can **generate** properties
  - Can not necessarily show more **specific behaviour**
    - *e.g., concurrency - which interleavings are possible*
- Based on the **input model**
  - What do we want to test and show with the model?

How to Create Properties?

- Based on **coverage criteria** (Testing with Model Checking [4])
  - State, transition, decision – even **path coverage** [5]
  - Can **generate** properties
  - Can not necessarily show more **specific behaviour**
    - *e.g., concurrency - which interleavings are possible*
- Based on the **input model**
  - What do we want to test and show with the model?
  - Can not fully **automate**

How to Create Properties?

- Based on **coverage criteria** (Testing with Model Checking [4])
  - Statement, decision, state, transition – even **path coverage** [5]
  - Can **generate** properties
  - Can not necessarily show more **specific behaviour**
    - *e.g., concurrency - which interleavings are possible*
- Based on the **input model**
  - What do we want to test and show with the model?
  - Can not fully **automate**

How do we know that we did not **miss** any wrong behaviour?

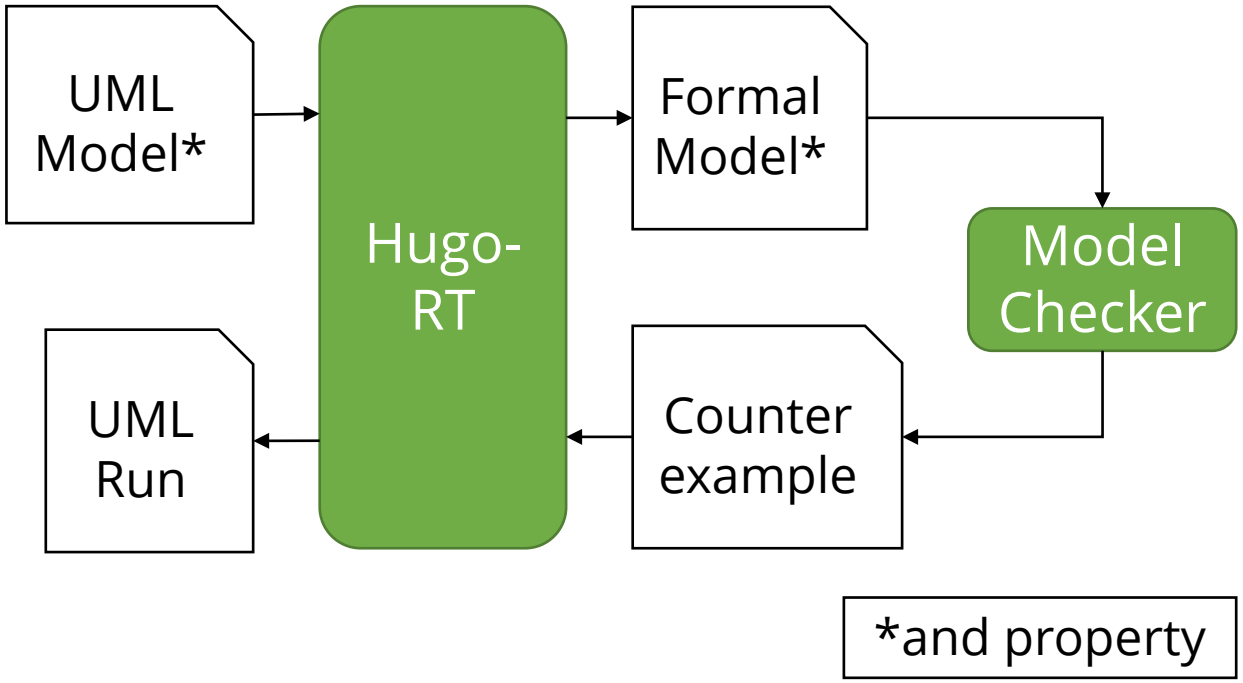
Case Study: Hugo-RT

„A UML model translator for model checking and code generation”

https://www.uni-augsburg.de/en/fakultaet/fai/informatik/prof/swtsse/hugo-rt/

Case Study: Hugo-RT

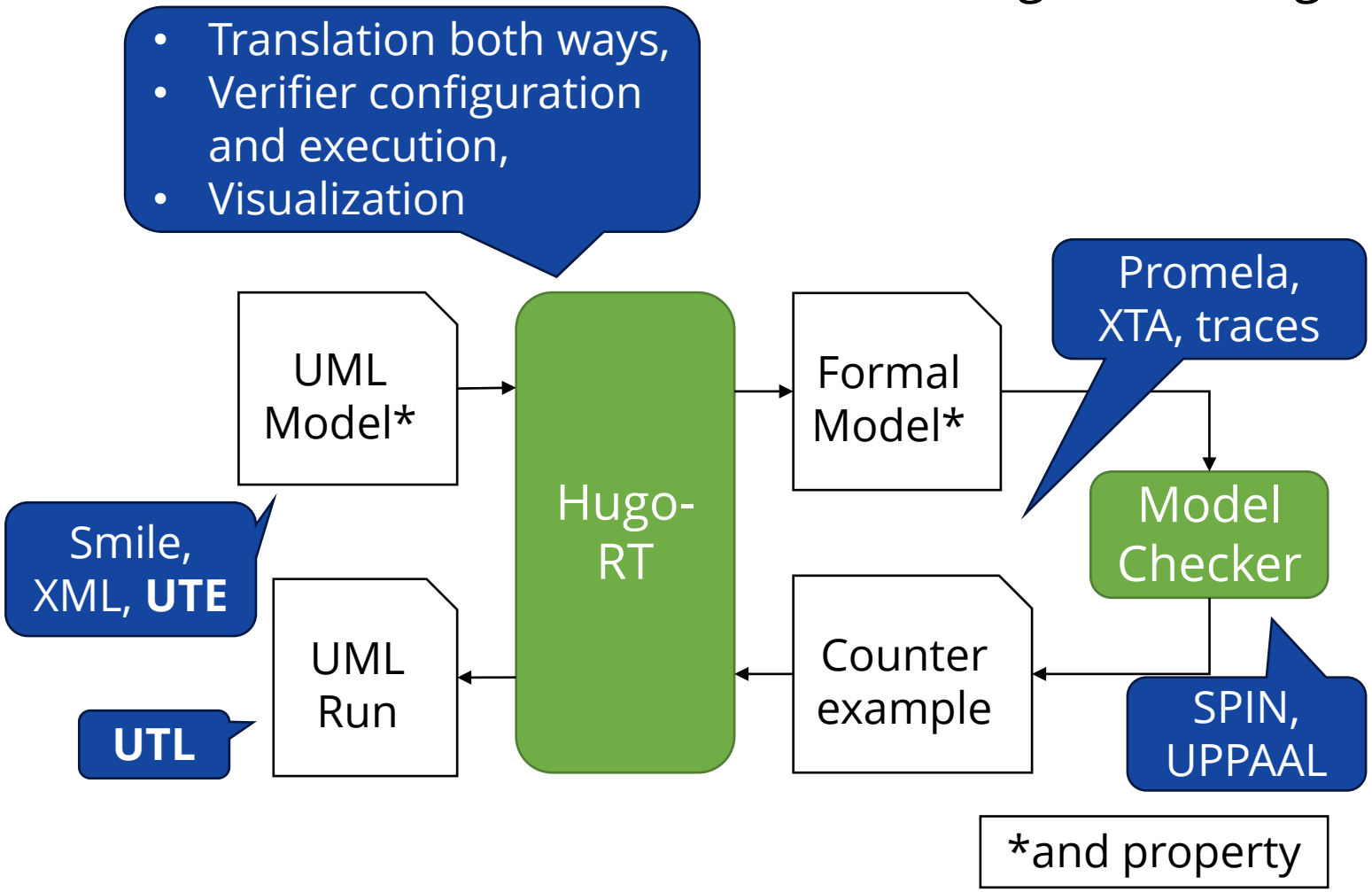
„A UML model translator for model checking and code generation”



https://www.uni-augsburg.de/en/fakultaet/fai/informatik/prof/swtsse/hugo-rt/

Case Study: Hugo-RT

„A UML model translator for model checking and code generation”

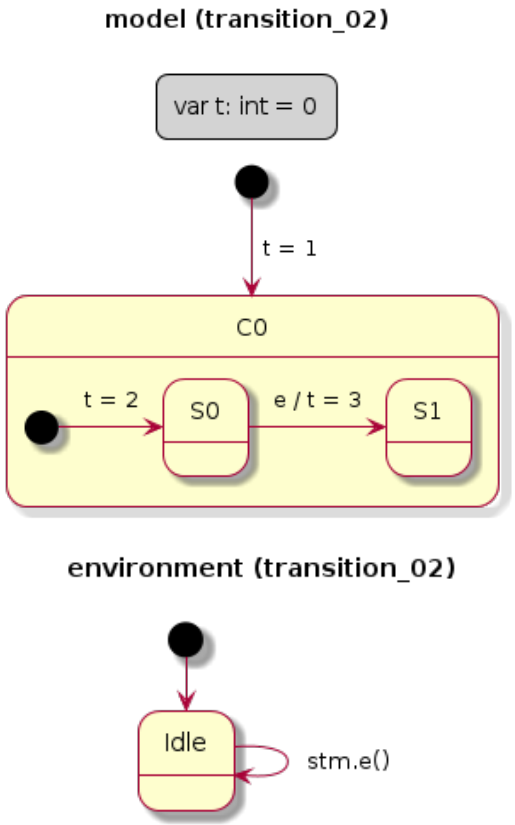


https://www.uni-augsburg.de/en/fakultaet/fai/informatik/prof/swtsse/hugo-rt/

Case Study: E2E Validation of Hugo-RT

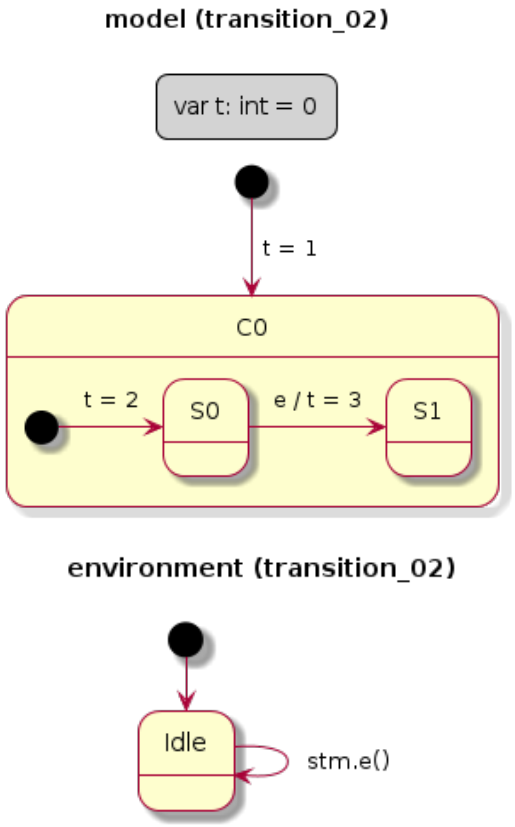
Case Study: E2E Validation of Hugo-RT

1, Validation models



Case Study: E2E Validation of Hugo-RT

1, Validation models
2, Generate properties *(parse statechart)*

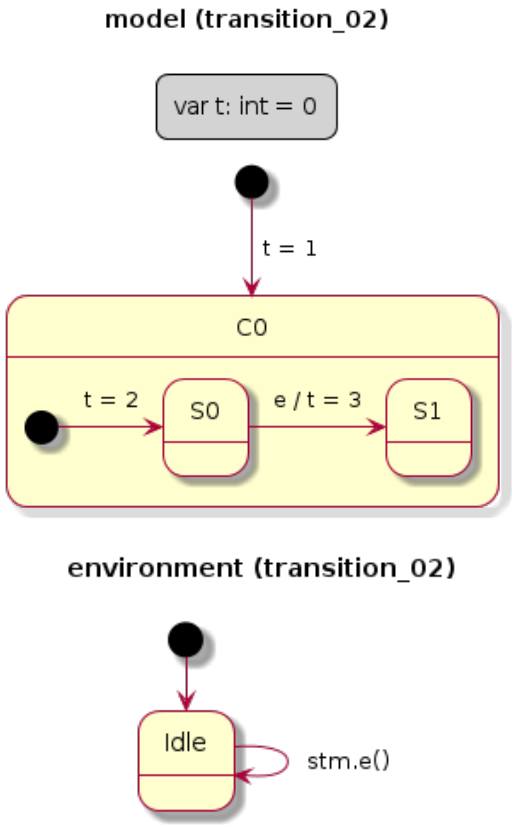



```
assertion reach_state_C0_S0 {
        G not stm1.inState(C0.S0);
}
assertion reach_state_C0_S1 {
        G not stm1.inState(C0.S1);
}
assertion reach_transition_3 {
        G stm1.transition != 3;
}
```

Transition reachability not possible directly!

Case Study: E2E Validation of Hugo-RT

1, Validation models
2, Generate properties *(parse statechart)*
3, Generate traces *(parse traces to check coverage)*
4, Check traces



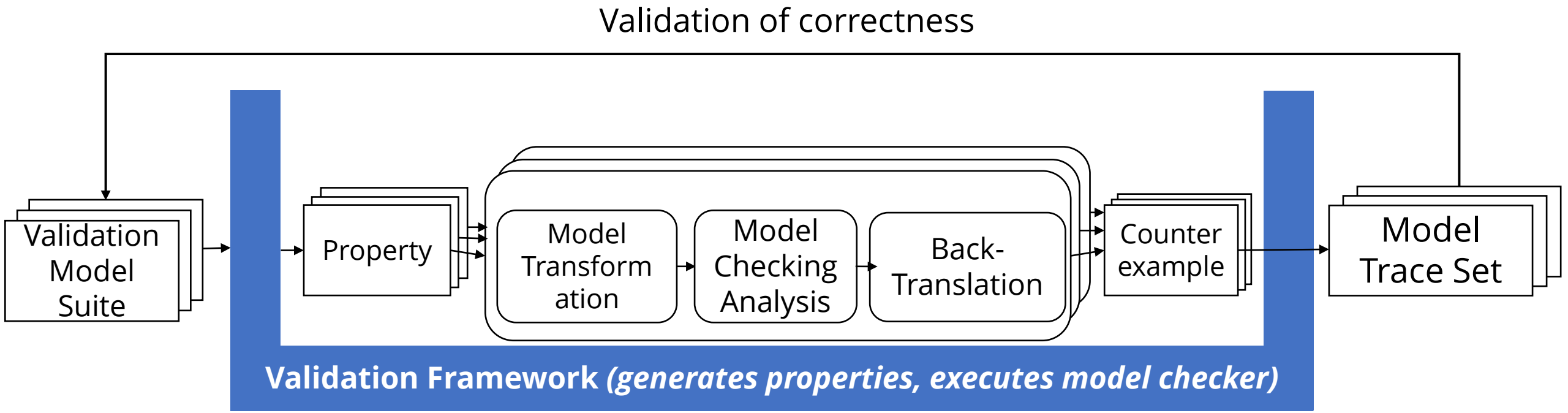

```
assertion reach_state_C0_S0 {
        G not stm1.inState(C0.S0);
}
assertion reach_state_C0_S1 {
        G not stm1.inState(C0.S1);
}
assertion reach_transition_3 {
        G stm1.transition != 3;
}
```

Transition reachability not possible directly!

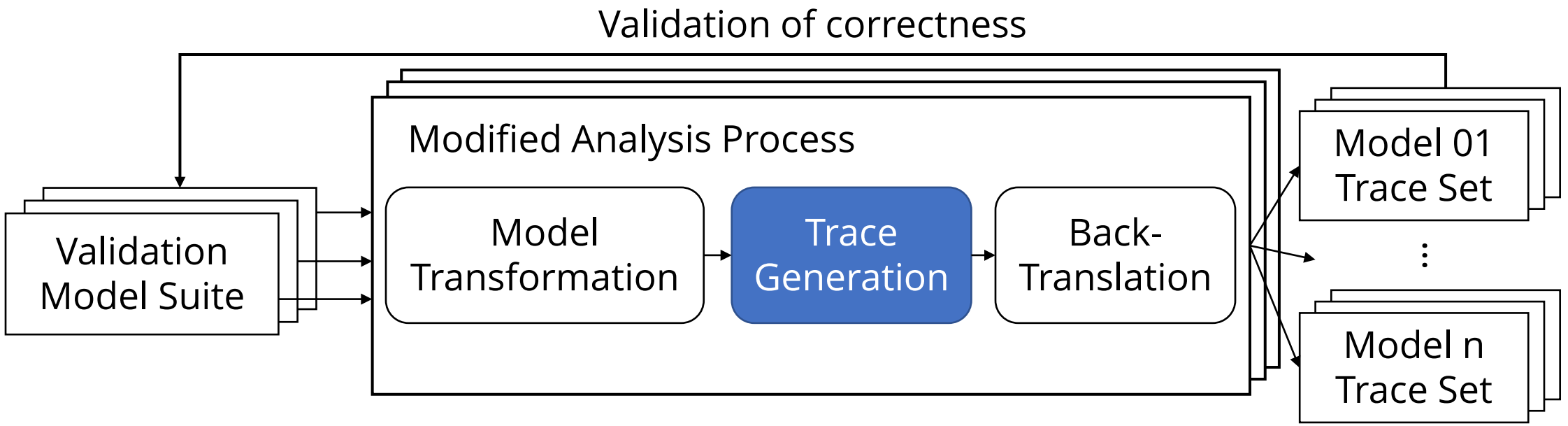
Coverage Goal	Covered	# traces
State Coverage	S0, S1 *(2/2)*	1
Transition Coverage	t=3 *(1/1)*	1

Issues with Validation Method Proposal 1.

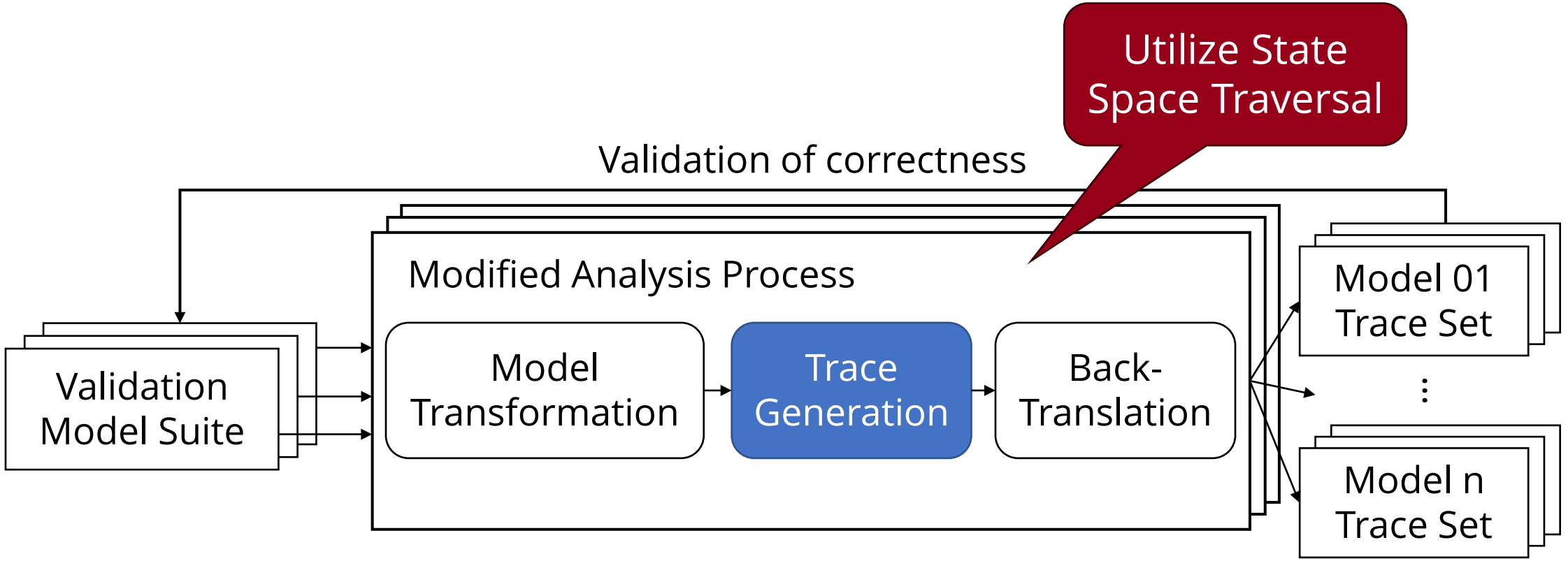
- How can we cover **all possible behaviors**?
- Model checker executed **for each property**
- **High effort** framework implementation



E2E Validation Method Proposal 2.



E2E Validation Method Proposal 2.



State Space-based Trace Generation

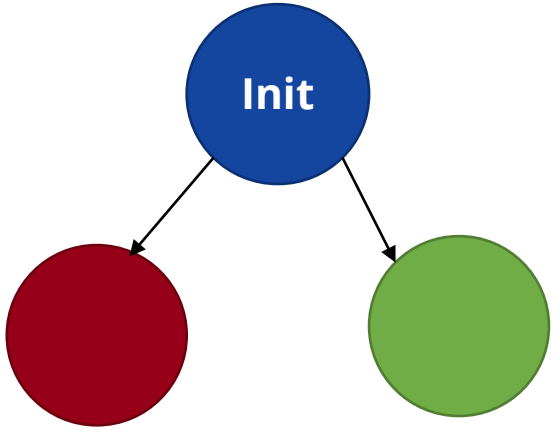
In the case of (Abstract) Reachability Graphs

State Space-based Trace Generation

In the case of (Abstract) Reachability Graphs

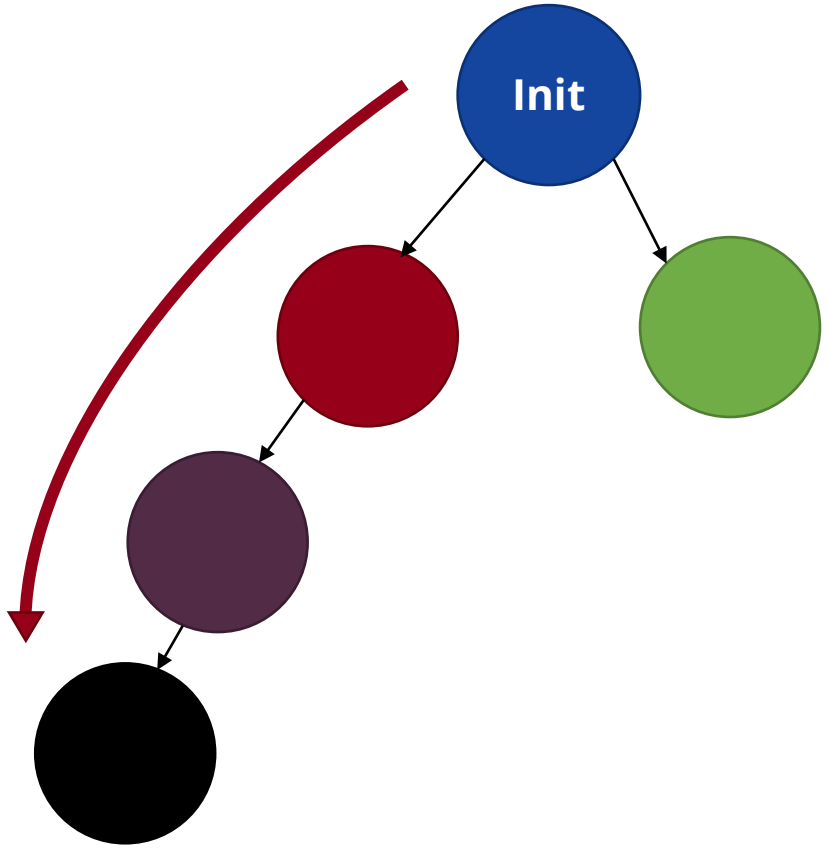
State Space-based Trace Generation

In the case of (Abstract) Reachability Graphs



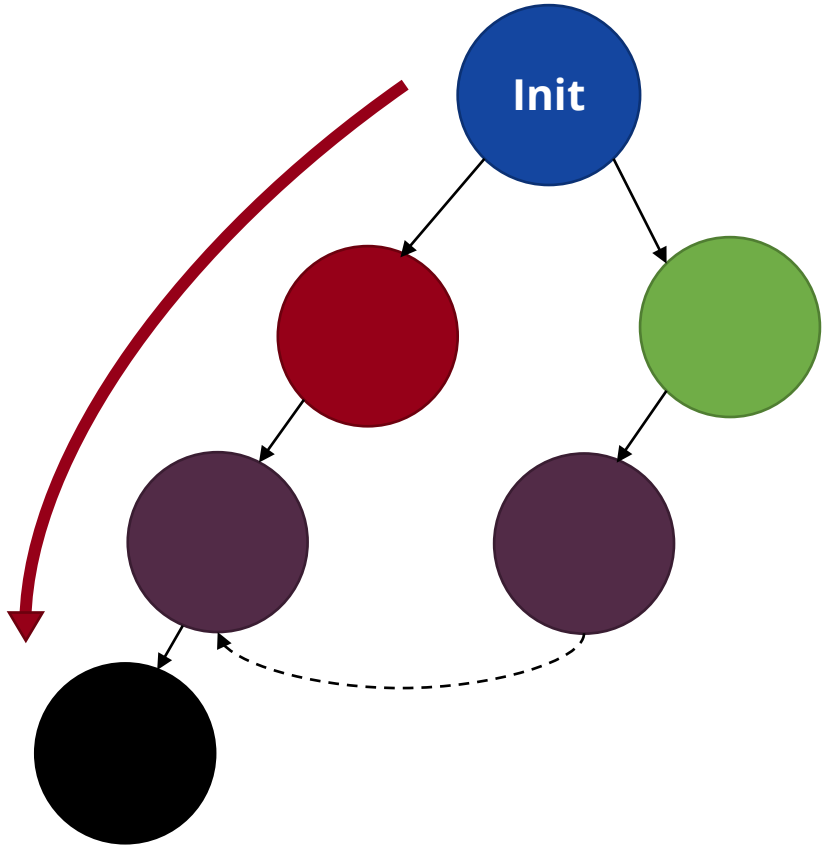
State Space-based Trace Generation

In the case of (Abstract) Reachability Graphs



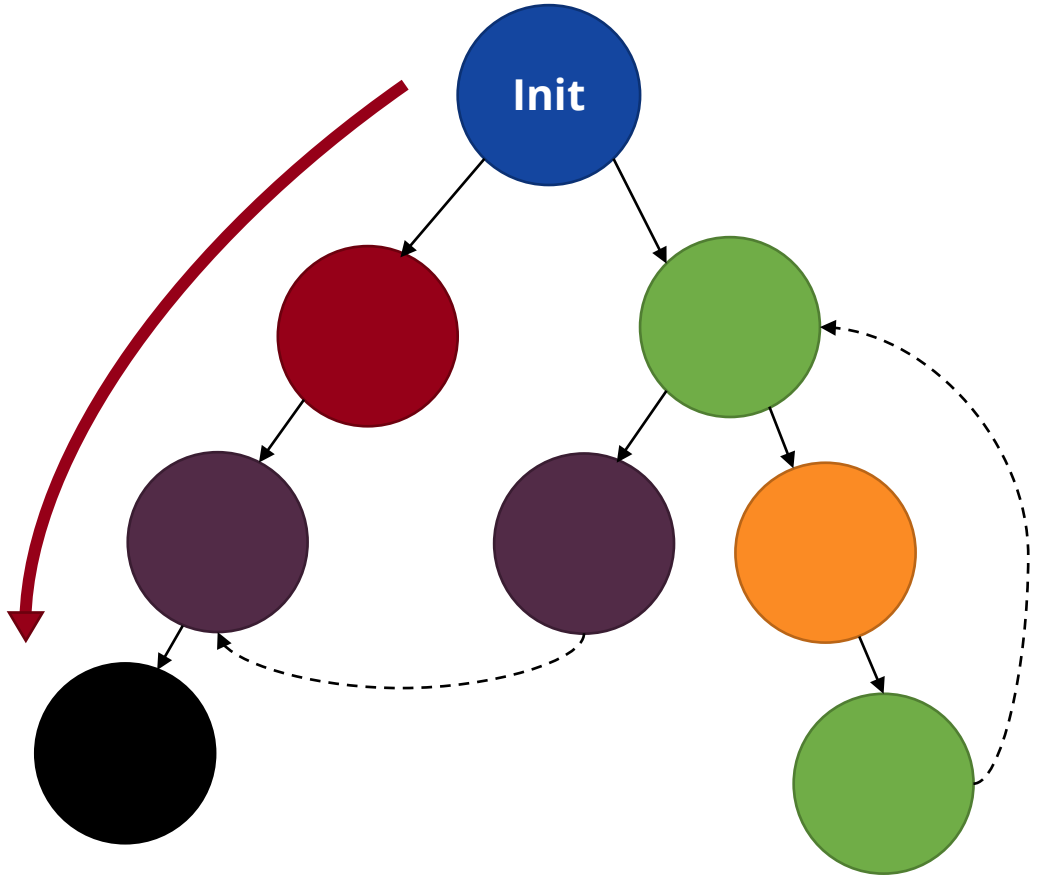
State Space-based Trace Generation

In the case of (Abstract) Reachability Graphs



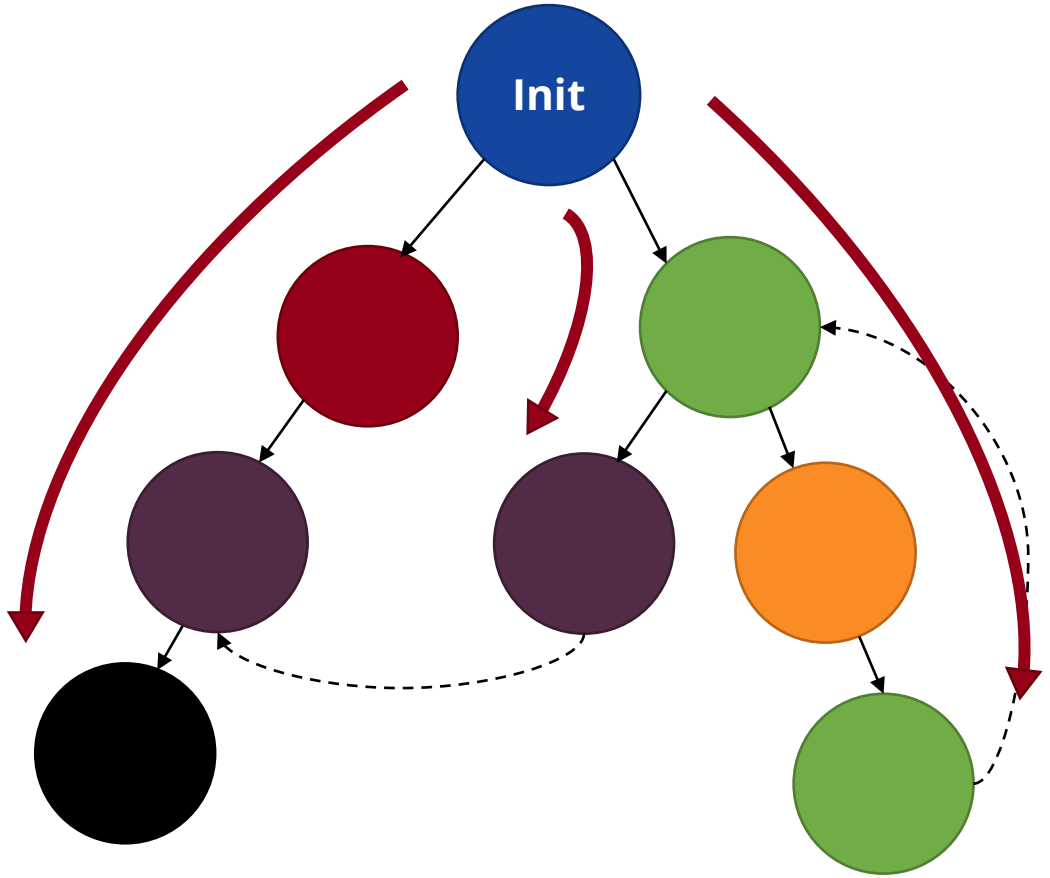
State Space-based Trace Generation

In the case of (Abstract) Reachability Graphs

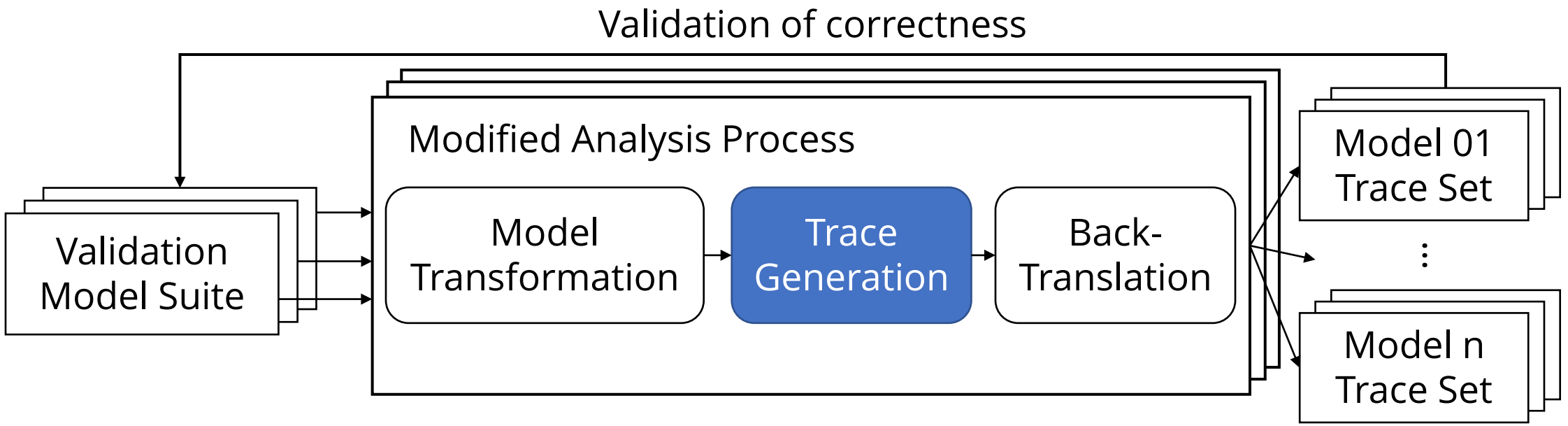


State Space-based Trace Generation

In the case of (Abstract) Reachability Graphs



E2E Validation Method Proposal 2.



Gamma+Theta Implementation

www.github.com/ftsrg/gamma
www.github.com/ftsrg/theta

Gamma+Theta Implementation

Prototype Implementation:
Gamma/Theta Toolchain

www.github.com/ftsrg/gamma
www.github.com/ftsrg/theta

Gamma+Theta Implementation

E2E Validation

Prototype Implementation:
Gamma/Theta Toolchain

www.github.com/ftsrg/gamma
www.github.com/ftsrg/theta

Gamma+Theta Implementation

NASA JPL, Prolan

E2E Validation

Real World Models

Prototype Implementation:
Gamma/Theta Toolchain

www.github.com/ftsrg/gamma
www.github.com/ftsrg/theta

Gamma+Theta Implementation

Generated Traces

- Mostly **1-4 traces** per model
- **Readable length** *(maximum 4)*

Findings

- Bugs, limitations
  - Missing default values
  - Unexpected ordering of operations

NASA JPL, Prolan

E2E Validation

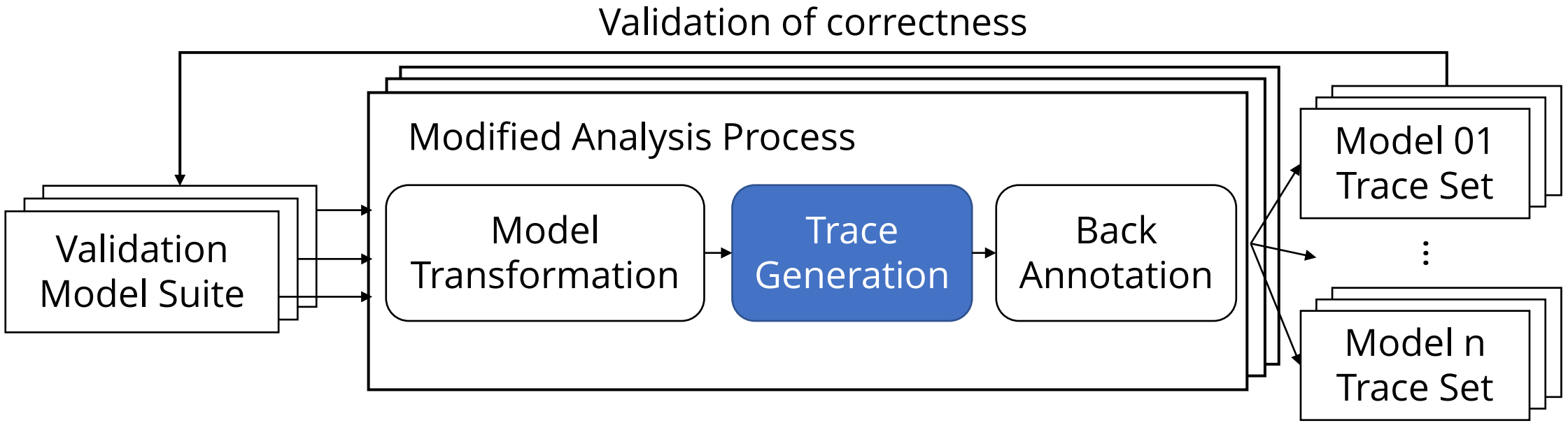
Real World Models

Prototype Implementation:
Gamma/Theta Toolchain

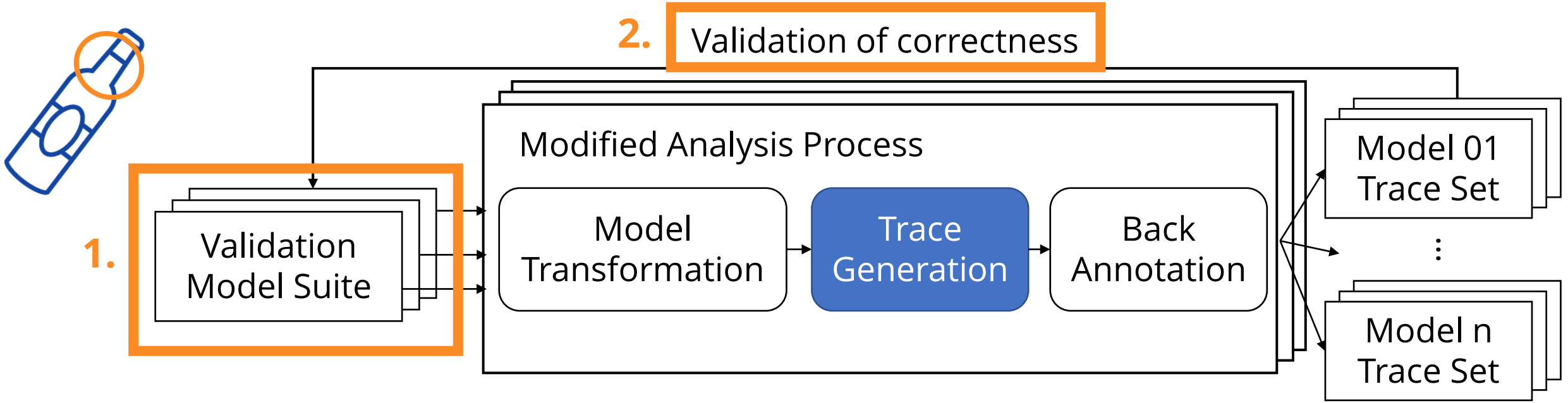
www.github.com/ftsrg/gamma
www.github.com/ftsrg/theta

Limitations and Ideas to Improve

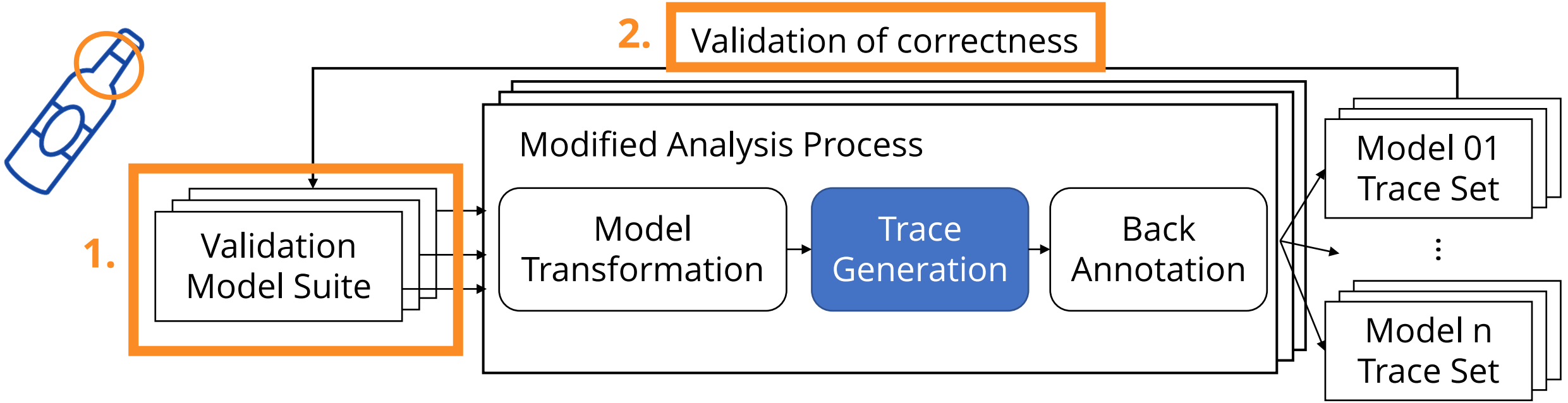
Validation Bottlenecks



Validation Bottlenecks

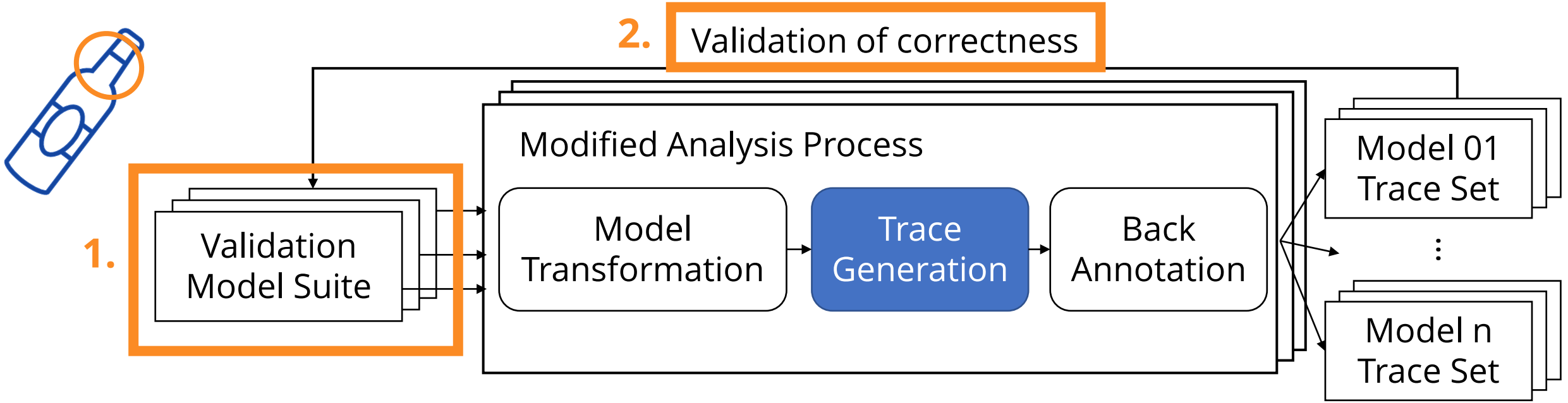


Validation Bottlenecks



- **Manual steps** – highly limiting
  - Reason: specification of input model language informal

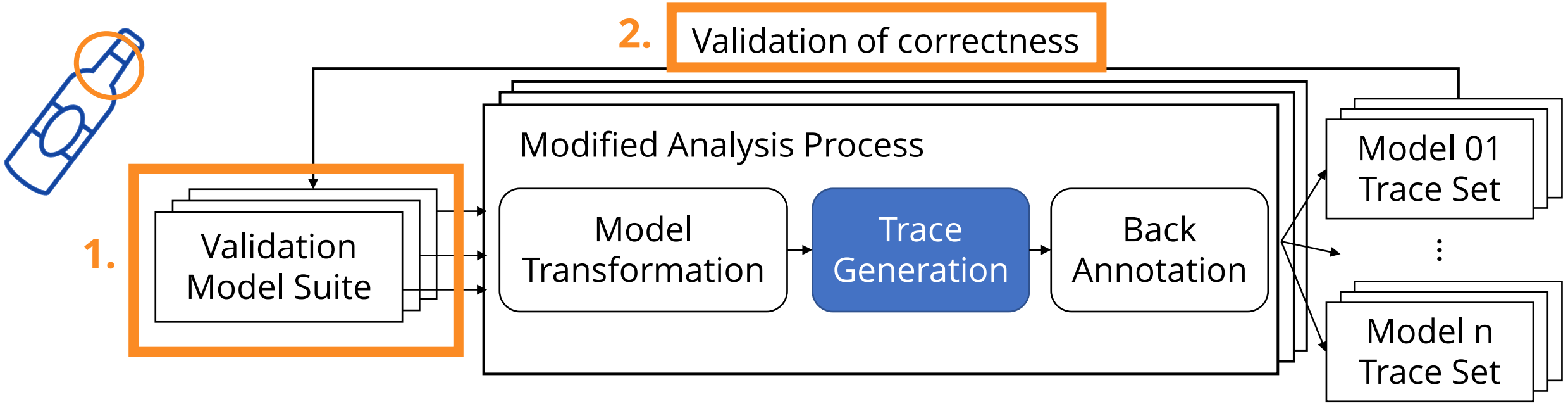
Validation Bottlenecks



- **Manual steps** – highly limiting
  - Reason: specification of input model language informal

1. Validation Suite: **Generate Models**

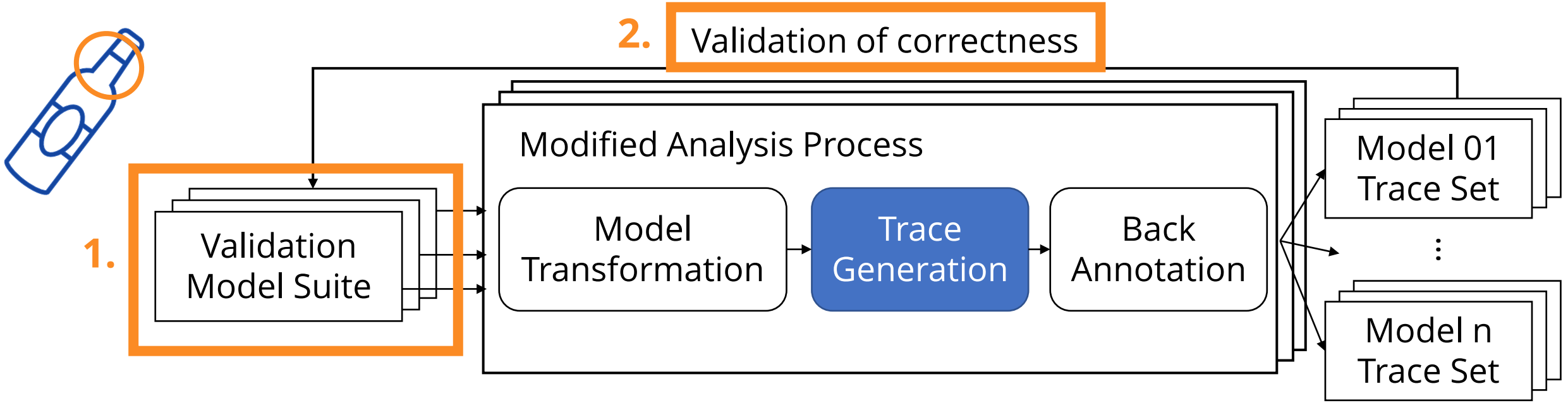
Validation Bottlenecks



- **Manual steps** – highly limiting
  - Reason: specification of input model language informal

1. Validation Suite: **Generate Models**
   - Refinery (refinery.tools)

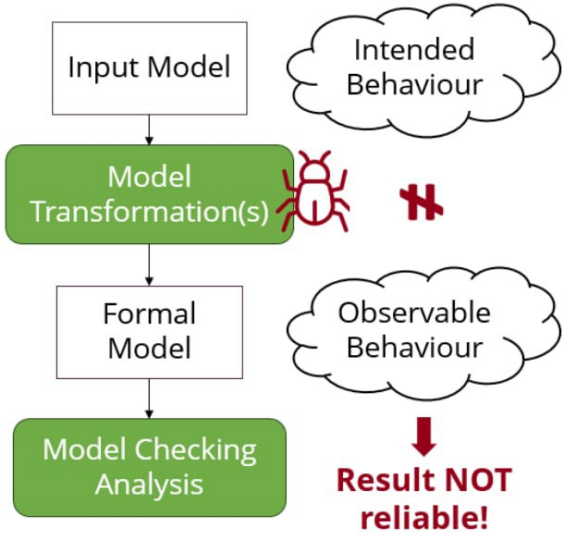
Validation Bottlenecks

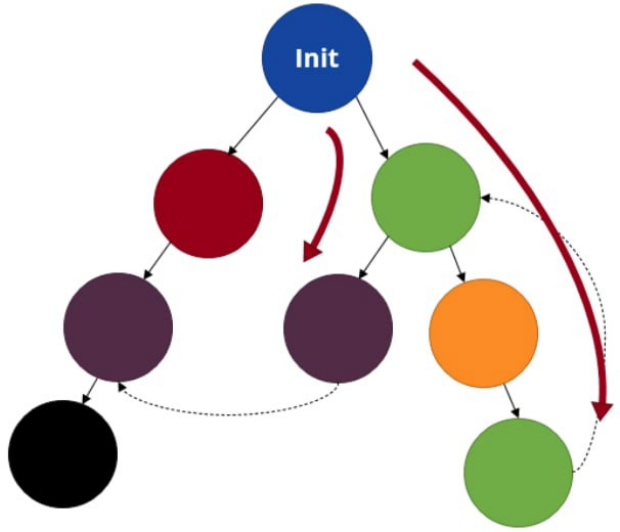


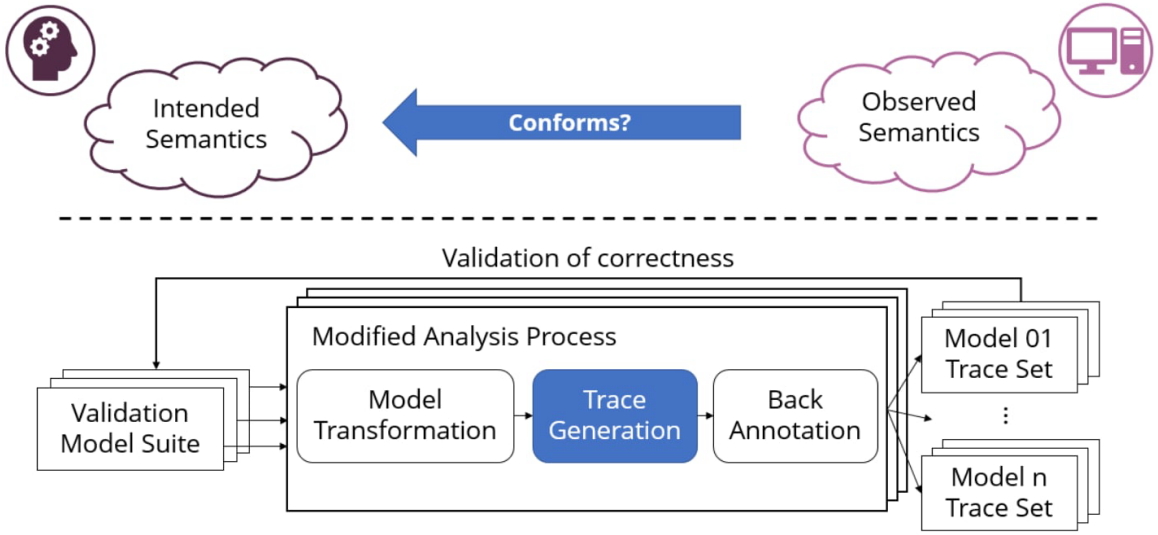
- **Manual steps** – highly limiting
  - Reason: specification of input model language informal

1. Validation Suite: **Generate Models**
   - Refinery (refinery.tools)
2. Validation of Correctness: some kind of **„summary"**
   - **Generate Automata** from Traces (“High-Level/Input Level (A)RG”)

Summary

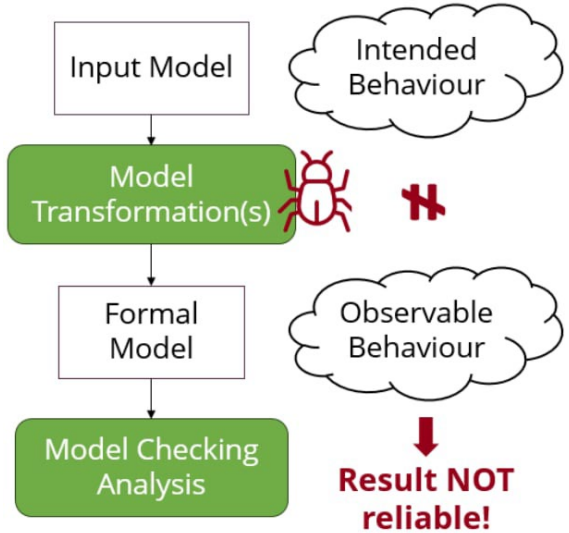

Input Model
Intended Behaviour
Model Transformation(s)
Formal Model
Observable Behaviour
Model Checking Analysis
Result NOT reliable!

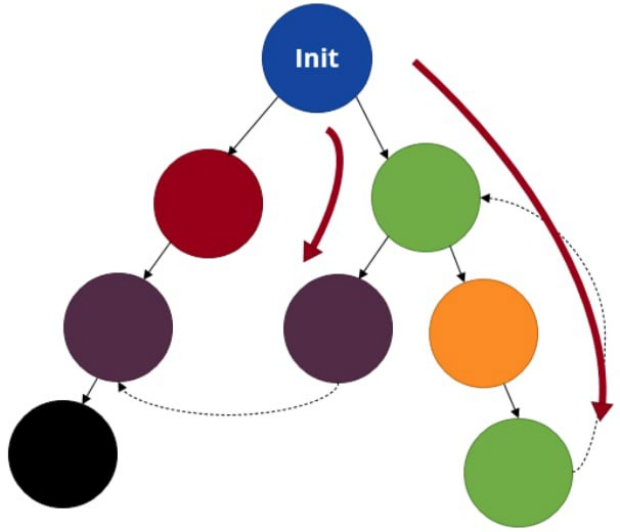

Init

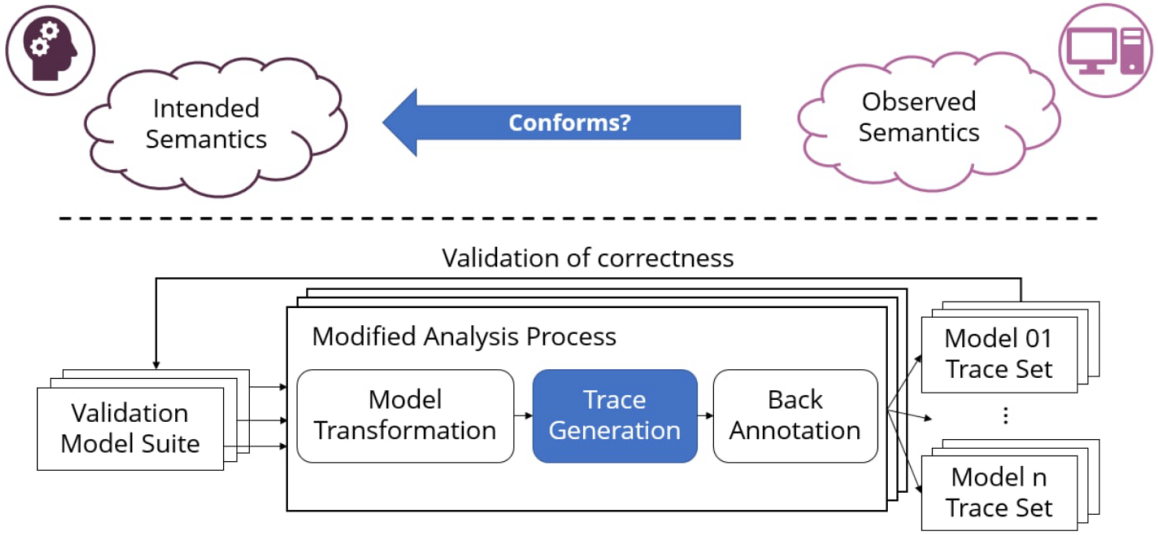

Intended Semantics
Conforms?
Observed Semantics
Validation of correctness
Modified Analysis Process
Validation Model Suite
Model Transformation
Trace Generation
Back Annotation
Model 01 Trace Set
Model n Trace Set

Summary

- **E2E validation** of model transformations in model checking toolchains

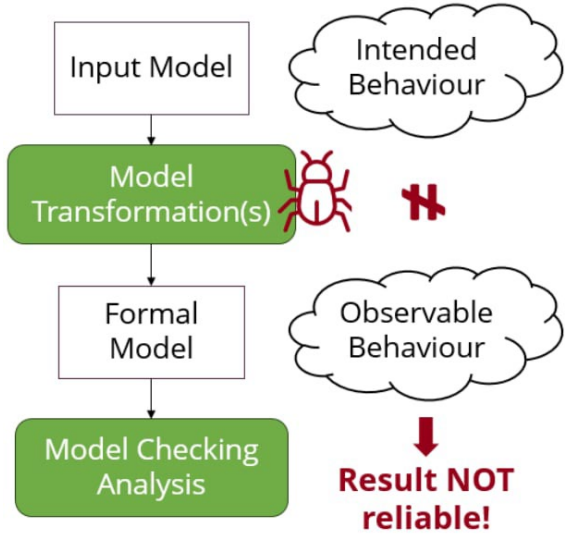


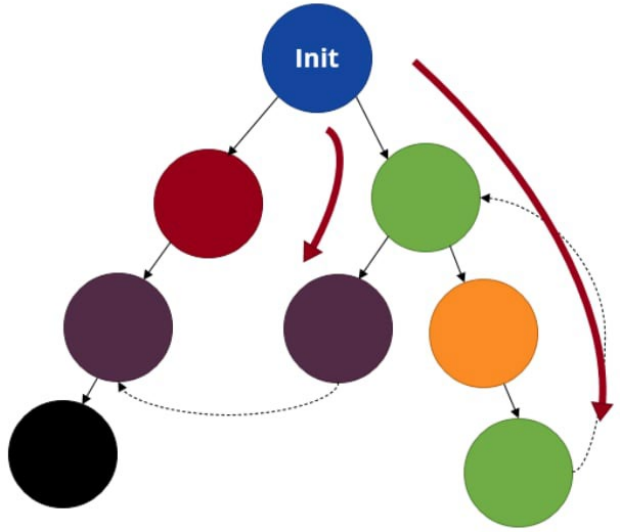


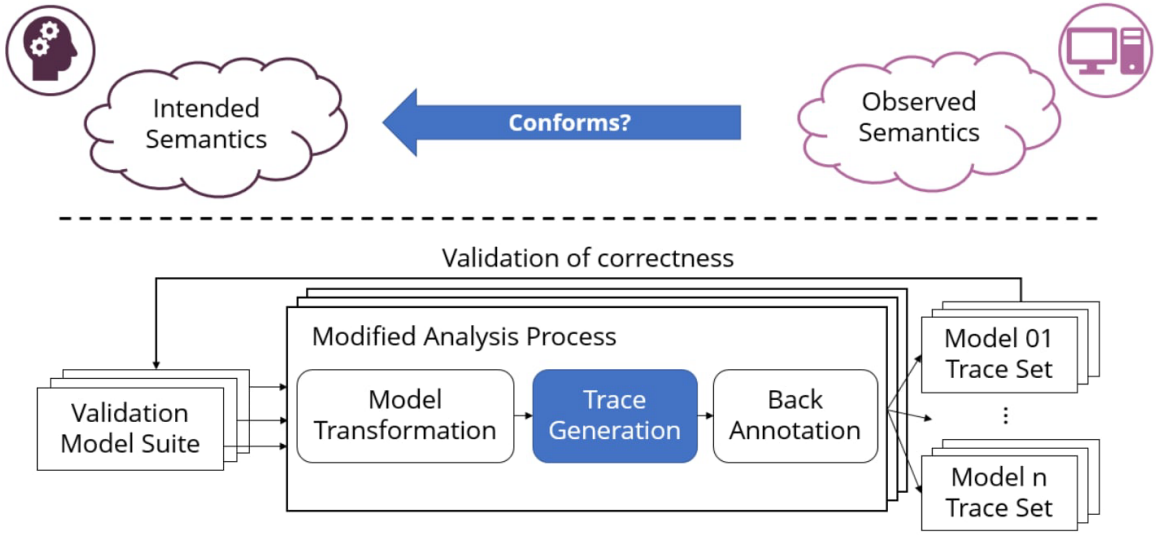


Summary

- **E2E validation** of model transformations in model checking toolchains
- Utilize **state space traversal**

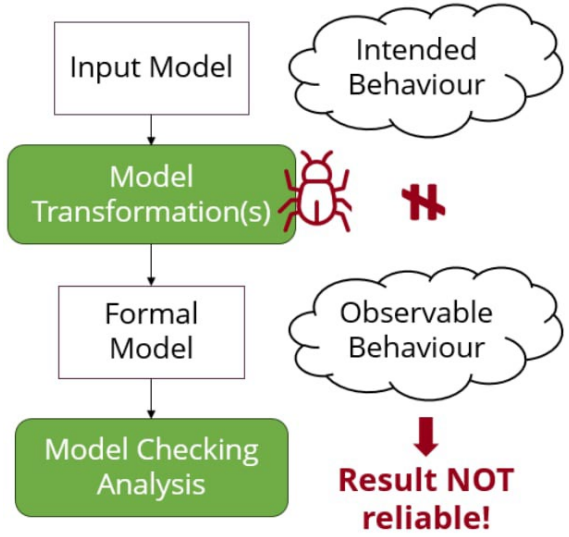


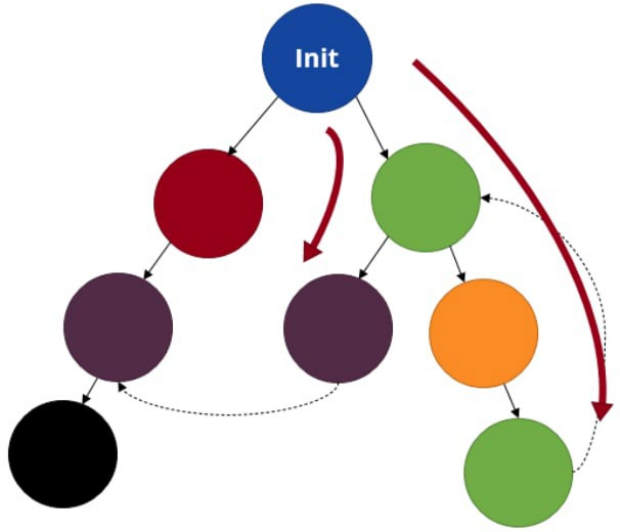


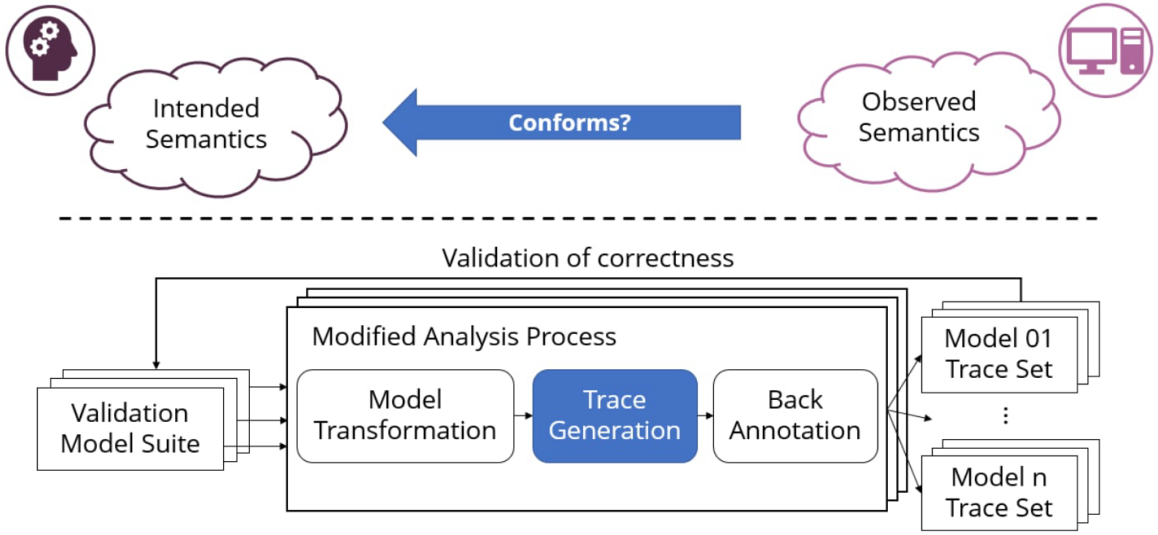


Summary

- **E2E validation** of model transformations in model checking toolchains
- Utilize **state space traversal**
- **Next**: generate validation model suites and behaviour summaries

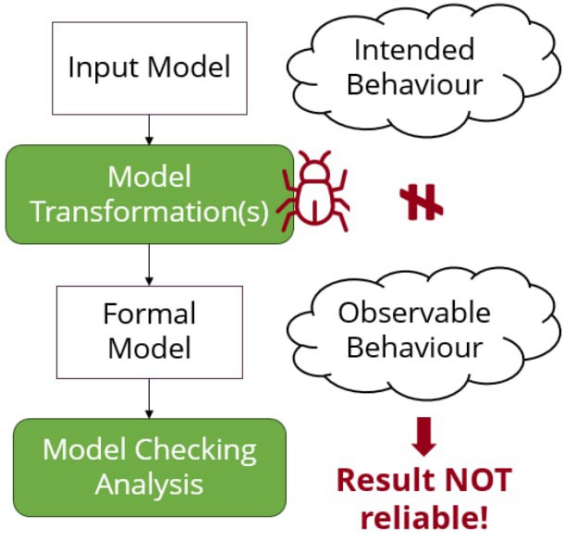


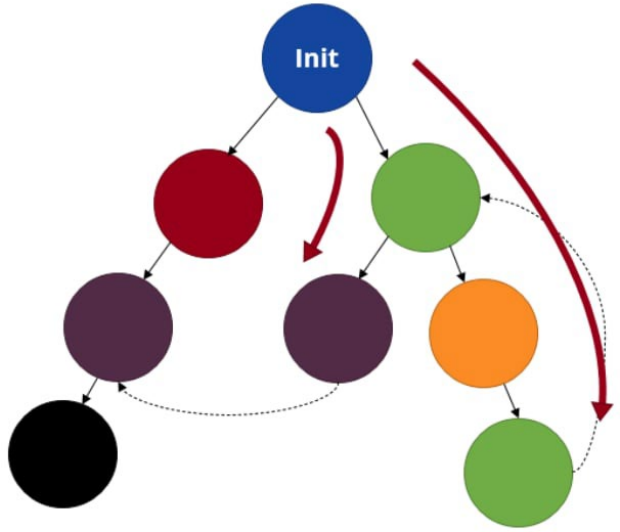


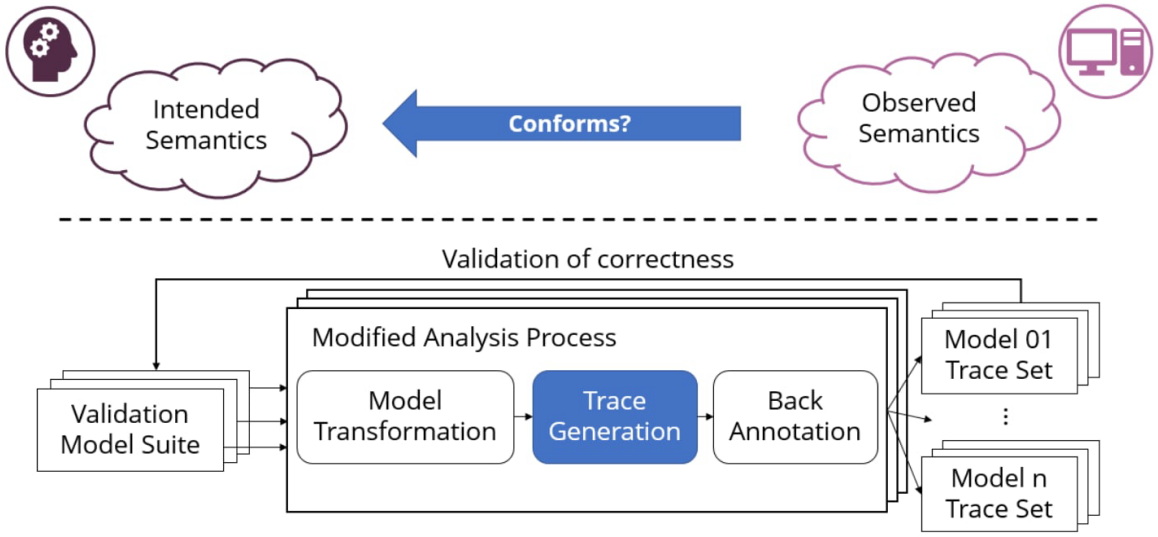


Summary

- **E2E validation** of model transformations in model checking toolchains
- Utilize **state space traversal**
- **Next**: generate validation model suites and behaviour summaries
- Looking for more **Ideas, Tools to Validate and Use Cases**

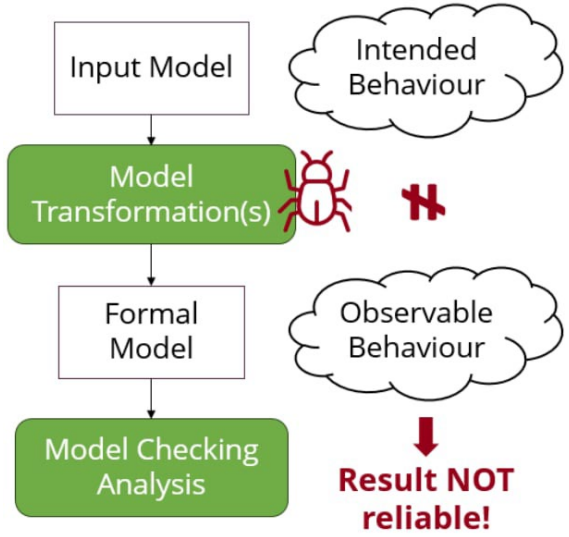


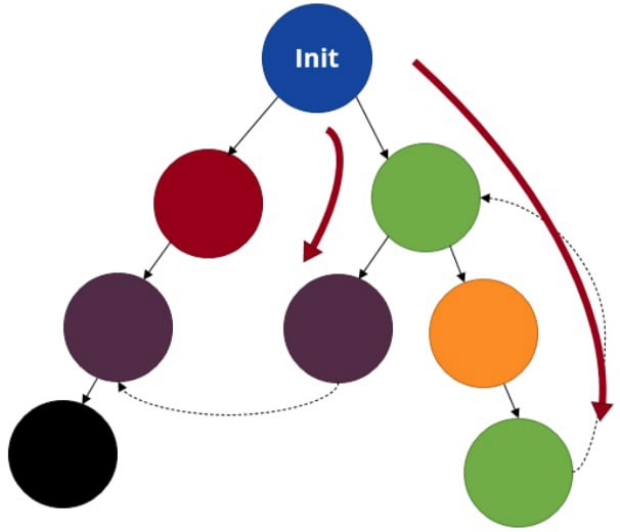


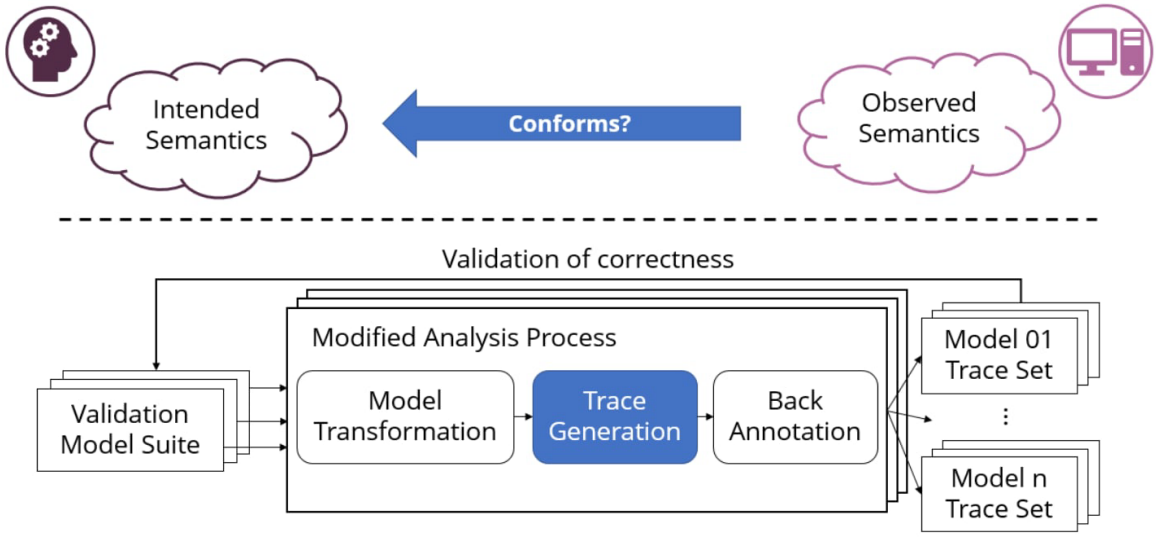


Summary

- **E2E validation** of model transformations in model checking toolchains
- Utilize **state space traversal**
- **Next**: generate validation model suites and behaviour summaries
- Looking for more **Ideas, Tools to Validate and Use Cases**
- *Where would you use this?*







Bibliography

[1] Varró, D., et al., Automated formal verification of visual modeling languages by model checking, https://doi.org/10.1007/s10270-003-0050-x

[2] Pretschner, A. et al., A Generic Fault Model for Quality Assurance, https://doi.org/10.1007/978-3-642-41533-3_6

[3] Chen, J., et al., A Survey of Compiler Testing, https://doi.org/10.1145/3363562

[4] Fraser, G. et al., Testing with model checkers: a survey, https://doi.org/10.1002/stvr.402

[5] Groce, A., (Quickly) testing the tester via path coverage, https://doi.org/10.1145/2134243.2134249