ThEodorE: a Trace Checker for CPS Properties

Claudio Menghi
University of Luxembourg
Luxembourg, Luxembourg
claudio.menghi@uni.lu

Enrico Viganò
University of Luxembourg
Luxembourg, Luxembourg
enrico.vigano@uni.lu

Domenico Bianculli
University of Luxembourg
Luxembourg, Luxembourg
domenico.bianculli@uni.lu

Lionel C. Briand
University of Luxembourg
Luxembourg, Luxembourg
University of Ottawa
Ottawa, Canada
lionel.briand@uni.lu

***Abstract*—ThEodorE is a trace checker for Cyber-Physical systems (CPS). It provides users with (i) a GUI editor for writing CPS requirements; (ii) an automatic procedure to check whether the requirements hold on execution traces of a CPS. ThEodorE enables writing requirements using the Hybrid Logic of Signals (HLS), a novel, logic-based specification language to express CPS requirements. The trace checking procedure of ThEodorE reduces the problem of checking if a requirement holds on an execution trace to a satisfiability problem, which can be solved using off-the-shelf Satisfiability Modulo Theories (SMT) solvers. This artifact paper presents the tool support provided by ThEodorE.**

***Index Terms*—Monitors, Languages, Specification, Validation, Formal methods, Semantics**

I. Introduction

A common practice to check whether a system behaves as expected is to collect and analyze execution traces. Traces are sequences of records that contain information about the state of the system variables. Each record is usually labeled with a time-stamp, i.e., the time instant at which the record was obtained. Traces are collected from simulations or during the actual system execution. They are then analyzed by engineers, to check whether they conform to the system's requirements.

Trace-checking tools automatically verify whether traces conform to system's requirements. Specification-driven trace-checking tools are a particular class of trace-checking tools that take as input a trace to be analyzed and a requirement specification, i.e., a property. They yield a Boolean verdict indicating whether the trace satisfies (or violates) the property.

ThEodorE [1, 2] is a specification-driven trace-checker for CPS properties. Figure 1 presents an overview of the main components of ThEodorE: *ThEodorE-GUI* and *ThEodorE-checker.*

ThEodorE-GUI allows engineers to express requirements using the *Hybrid Logic of Signals (HLS)* [3], a new specification language tailored to specifying CPS requirements. HLS allows engineers to express CPS requirements as properties that refer both to the time-stamps and the indices of the trace records. In this way, HLS properties can easily express the behavior of both cyber and physical components, as well as their interactions.

ThEodorE-checker provides an efficient trace-checking procedure for properties expressed in HLS. ThEodorE reduces the problem of checking an HLS property on a trace to a satisfiability problem, which can be solved using off-the-shelf Satisfiability Modulo Theories (SMT) solvers. SMT solvers have efficient decision procedures for several background theories, making it possible to check whether a formula is satisfiable.

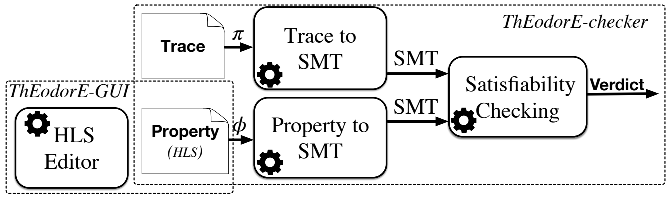


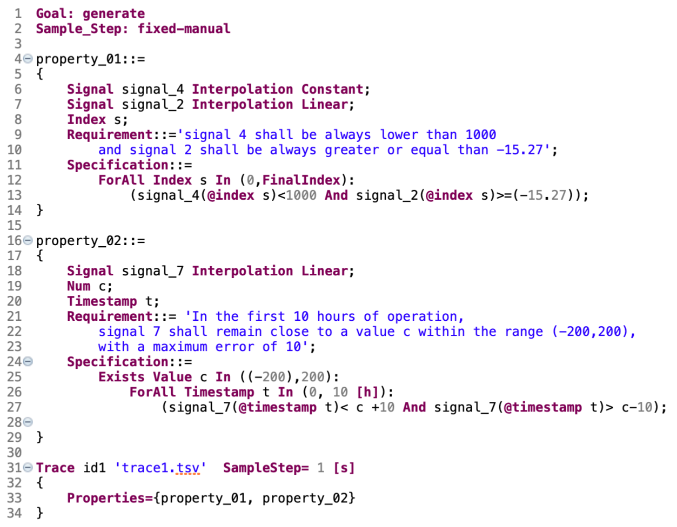
Figure 1: The main components of ThEodorE.

In the following, we provide an overview of the tool support provided by *ThEodorE-GUI* and *ThEodorE-checker.*

II. ThEodorE-GUI

ThEodorE-GUI is implemented as an Eclipse plugin using Xtext [4]. Figure 2 presents a screenshot of the ThEodorE-GUI editor showing two requirements specified as properties: `property_01` and `property_02`. Each property is composed of:

- *Requirement* (**`Requirement`**): a textual description of the requirement captured by the property, included for documentation purposes.
- *Signals definition* (**`Signal`**): the definition of the signal identifiers used in the HLS specification, and the interpolation functions used to generate the missing values of each signal in the trace. ThEodorE currently supports the **`Constant`** and **`Linear`** interpolation functions. When the value of a signal is missing in a record, the **`Constant`** interpolation assigns the missing value of a signal in a record to the value assigned to the signal in the previous record. The **`Linear`** interpolation assigns the missing value of a signal in a record to a new value computed by connecting, with a hypothetical straight line, the values assigned to the signal in the previous and in the next record.
- *Variables definition* (**`Index`**, **`Timestamp`**, **`Num`**): the definition of the timestamp, index and real-valued variables used in the HLS specification;



```
 1  Goal: generate
 2  Sample_Step: fixed-manual
 3
 4  property_01::=
 5  {
 6      Signal signal_4 Interpolation Constant;
 7      Signal signal_2 Interpolation Linear;
 8      Index s;
 9      Requirement::='signal 4 shall be always lower than 1000
10          and signal 2 shall be always greater or equal than -15.27';
11      Specification::=
12          ForAll Index s In (0,FinalIndex):
13              (signal_4(@index s)<1000 And signal_2(@index s)>=(-15.27));
14  }
15
16  property_02::=
17  {
18      Signal signal_7 Interpolation Linear;
19      Num c;
20      Timestamp t;
21      Requirement::= 'In the first 10 hours of operation,
22          signal 7 shall remain close to a value c within the range (-200,200),
23          with a maximum error of 10';
24      Specification::=
25          Exists Value c In ((-200),200):
26              ForAll Timestamp t In (0, 10 [h]):
27                  (signal_7(@timestamp t)< c +10 And signal_7(@timestamp t)> c-10);
28
29  }
30
31  Trace id1 'trace1.tsv'  SampleStep= 1 [s]
32  {
33      Properties={property_01, property_02}
34  }
```

Figure 2: A screenshot of the ThEodorE-GUI editor.

- *The HLS formula* (**`Specification`**): the specification of the property of interest in HLS [3].

The code shown in Figure 2 indicates that properties `property_01` and `property_02` should be checked on the trace (**`Trace`**) stored in file `trace1.tsv`, which is associated with the identifier `id1`.

The ThEodorE-GUI also allows the user to specify the pre-processing strategies used by ThEodorE (see [3]). Specifically, the value assigned to **`Sample_Step`** (line 2) defines how to pre-process the original trace:

- **`fixed-manual`** indicates that the fixed sample step specified for each trace should be considered for pre-processing. For example, the sample step to be considered for pre-processing the trace `id1` is one second (see **`SampleStep`** at line 31).
- **`fixed-min`** indicates that the sample-step to be used for pre-processing is the smallest sample-step among the sample-steps of the records of the original trace.
- **`variable`** indicates that the same samples as in the original trace are considered.

III. THEODORE-CHECKER

ThEodorE-checker (see Figure 1) takes as input a property ϕ expressed in HLS and a trace π. The usage workflow of the tool includes the following steps:

- Translating the property ϕ and trace π into formulae expressed using the input language of the selected SMT solver. The translation is implemented in Xtend [5], a flexible and expressive dialect of Java. The translation is automatically triggered by (i) changing the value assigned to the **`Goal`** option (line 1 in Figure 2) from **`save`**, the value used while editing the requirements, to **`generate`**, and (ii) saving the file. The translation produces a Python file for each trace-requirement combination; the generated file uses the Z3 Python API for invoking the SMT solver.

```
python3 property_01_id1.py
sat
--- 0.8608121819482--- 0.8608121871948242 seconds ---
REQUIREMENT VIOLATED
```

Figure 3: An example output produced by the *ThEodorE-checker*.

- Verifying the satisfiability of the SMT formula ψ, obtained from the trace-requirement combination of interest at the end of the previous step, using the SMT solver. The satisfiability of ψ is checked by executing the Python file generated as part of the first step; the verification verdict is generated when the execution ends. For example, as shown in Figure 3, the satisfiability of the SMT formula corresponding to the trace-requirement combination of trace `id1` and the requirement specified in `property_01` is verified by running `python3 property_01_id1.py`, which is produced in the first step.

The final verdict yielded by *ThEodorE-checker* can be "*satisfied*", "*violated*" or "*unknown*". *ThEodorE-checker* yields the *definitive verdicts* "satisfied" or "violated" when the solver returns "UNSAT" or "SAT", indicating, respectively, that ψ is unsatisfiable or satisfiable. However, the solver may return an "UNKNOWN" answer since the satisfiability of the underlying logic used within the solver is generally undecidable. In our case, this indicates that no conclusion is drawn on the satisfiability of formula ψ, thus resulting in an "unknown" verdict returned by *ThEodorE-checker*.

For example, Figure 3 shows a fragment of the output of *ThEodorE-checker*, when checking property `property_01` on trace `id1`; notice that the obtained verdict is "VIOLATED".

ACKNOWLEDGEMENT

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 694277), and NSERC of Canada under the Discovery and CRC programs.

REFERENCES

[1] "ThEodorE," https://github.com/SNTSVV/ThEodorE, 2020.

[2] C. Menghi, E. Viganò, D. Bianculli, and L. C. Briand, "Theodore - trace-checker," Feb. 2021. [Online]. Available: https://doi.org/10.5281/zenodo.4506795

[3] C. Menghi, E. Viganò, D. Bianculli, and L. C. Briand, "Trace-checking CPS properties: Bridging the cyber-physical gap," in *International Conference on Software Engineering (ICSE)*. ACM, 2021.

[4] "Xtext," https://www.eclipse.org/Xtext/, 2020.

[5] "Xtend," https://www.eclipse.org/xtend/, 2020.