CaRTLola

Reactive Systems Group

1 Project Overview

The central goal of the project is to develop a smartphone app that connects to the on-board network of a car and monitors the available data for safety-critical situations. The app implements a monitor that carries out an analysis that is specified in the formal specification language RTLola. Unlike specifications in a natural language, RTLola has a formal semantics with no ambiguities. RTLola is comparable to a programming language, but comes with much stronger guarantees on its runtime behaviour and use of resources such as memory.

Initially, our project will focus on the on-board diagnostics (OBD) interface, which provides incoming data such as speed, emission data and throttle. This data is forwarded to the developed smartphone application. Later, the project can integrate additional data sources. For example, analyzing images of the built-in camera of the smartphone allows for detecting traffic signs. This enables the monitor to check for speed limits.

The project is separated into three parts: The first part focuses on describing properties for safe driving behavior in natural language and later in RTLola. Next, you should implement a smartphone application for the RTLola API with the OBD interface; the choice of operating system is still open for debate. In the last part the app is evaluated in

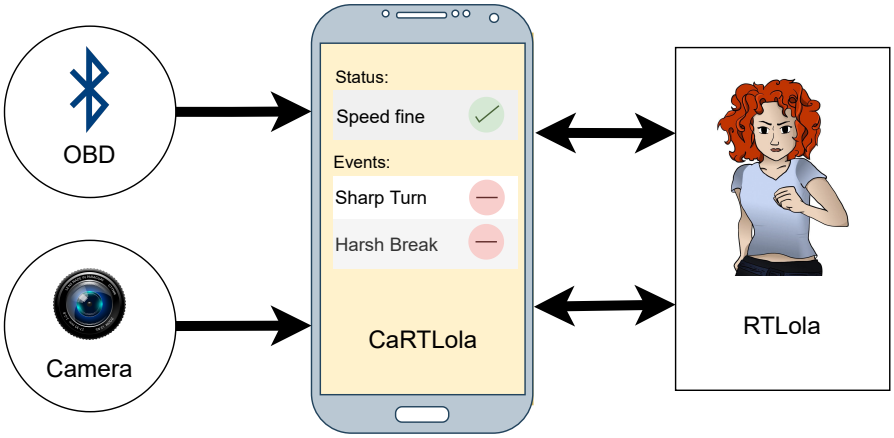


Figure 1: A rough sketch of the user interface of the App and how it interacts with the different components.

real driving scenarios. In particular this requires a comparison to commercially available products.

2 Technology Overview

In the following section we give an overview over the technologies that are going to be used in this project.

2.1 RTLola

RTLola [1] is a specification language designed to specify the behavior of a system in a clear and structured manner. For this, data is represented in infinite lists, so-called streams. RTLola differentiates between three types of streams: Input streams represents the data provided to the monitor. Output streams use stream equations to filter incoming data, compare values from different streams, or carry out more complex computations. Then, trigger streams characterize critical situations with boolean expressions. When a critical situation is detected, a message indicating the failure is emitted.

Consider the following example to gain an intuition of the language. This specification describes a simplified property checking if the driver accelerates too fast:

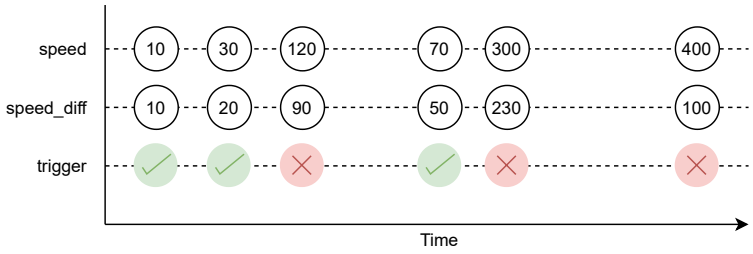
```
input speed: Int64

output speed_diff: Int64 := abs(speed - speed.offset(by: -1).defaults(to: 0))
```

```
trigger speed_diff > 50 "acceleration too high"
```

The specification contains three streams: The first stream declares the input `speed` representing the current velocity of the vehicle. The second stream then computes the difference between two consecutive speed values. For this, the stream equation takes the current value of the input stream, expressed by `speed`, and the previous input with the offset operator `speed.offset(by: -1)`. This operator accesses an xth past value of a stream if x is negative or the xth future value if x is positive. These values are not always present, i.e., in the beginning of the evaluation there is no previous value. RTLola resolves this issue with the default operator replacing non-present values. The last stream is a trigger declaration notifying the user if the value of the stream `speed_diff` is larger than a specified bound.

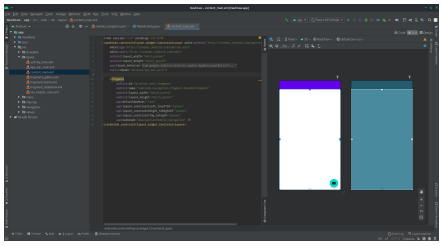
The evaluation process is sketched in the following figure:

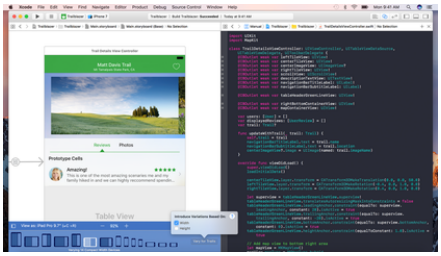


2.2 On-Board Diagnostics

Modern cars are equipped with many sensors providing technical support for the driver. The location of these sensors does not always coincide with the location where the data is processed. For the communication between these locations, a car uses a shared wire, usually a CAN-Bus, to save wiring and weight. Besides the sensors, an on-board diagnostic (OBD) [2] interface is connected to the bus forwarding the sensor data out of the closed system. This interface is typically used for diagnostic purposes and provides useful data including the current engine and vehicle speed, the throttle position, engine run time, and much more [3]. An OBD adapter provides this data over Bluetooth allowing the app to access it.

2.3 Developing Apps

 [4]

 [5]

Building smartphone applications has become significantly simpler over the past few years. This progress is mostly due to free IDEs like Android Studio [6] or XCode [7]. These IDEs features a graphical user interface editor and ship with an emulator to simulate a smartphone on your computer combining implementation, designing, and testing into a neat and easy to use workflow.

We don't expect that you are familiar with these development environments and learning everything you need is part of the project.

3 Milestones

We propose the following road map but are looking forward to your ideas.

Phase 1 — Specification

1. Specifying properties in natural language
2. Getting familiar with RTLola
3. Translating the properties from 1. into RTLola
4. Generating test data
5. Validating the specification on test data

Phase 2 — Implementation

6. Getting familiar with App Development
7. Designing the App frontend
8. Integrating the RTLola API into the App
9. Validating the App on test data

Phase 3 — Evaluation

10. Getting familiar with the OBD Interface
11. Integrating the OBD interface into the App
12. Testing the specification and App in a real-time environment
13. Adding road sign detection to the App / specification
14. Evaluating the App against commercial products

4 Apply Now!

To apply for participating in this project you should send an e-mail to:

rtlola-projekt@react.uni-saarland.de

References

[1] Rtlola sepcification language. `https://rtlola.org`.

[2] On-board diagnostics. `https://en.wikipedia.org/wiki/On-board_diagnostics`.

[3] Obd data. `https://en.wikipedia.org/wiki/OBD-II_PIDs`.

[4] By google llc and jetbrains screenshot:vulphere - self-taken; derivative work, apache license 2.0. `https://commons.wikimedia.org/w/index.php?curid=95495098`.

[5] By source (wp:nfcc4), fair use. `https://en.wikipedia.org/w/index.php?curid=53398394`.

[6] Android studio. `https://developer.android.com/studio`.

[7] Xcode. `https://developer.apple.com/xcode/`.