Implementation and Semantics of an asynchronous evaluation engine for stream based specifications

---

Alexander Schramm

November 20, 2016
Version: My First Draft

University of Luebeck

UNIVERSITÄT ZU LÜBECK

Institute For Software Engineering and Programming Languages

isp

Master Thesis

Implementation and Semantics of an asynchronous evaluation engine for stream based specifications

Alexander Schramm

1. Reviewer Prof. Dr. Martin Leucker
Institute For Software Engineering and Programming Languages
University of Luebeck

2. Reviewer Who Knöws
We will see
University of Luebeck

Supervisors Cesar Sanchez

November 20, 2016

Alexander Schramm

Implementation and Semantics of an asynchronous evaluation engine for stream based specifications

Master Thesis, November 20, 2016
Reviewers: Prof. Dr. Martin Leucker and Who Knöws
Supervisors: Cesar Sanchez

University of Luebeck
Institute For Software Engineering and Programming Languages
Ratzeburger Allee 160
23562 Luebeck

Abstract

Abstract (Deutsch)

Acknowledgement

Contents

1 System

Let E be the Set of valid input events and $e \in E$, where each event carries a value, a timestamp and the channel it is perceived on (e.g. a function call of a specific function). Further let O be the Set of valid output events and $o \in O$, which have the same properties than input events.

1.1 Semantics of TeSSLa functions

Think of semantics of thinks like add, mrv, etc. Can these be defined independent of async/synchronous or for each. For synchronous they are partly defined in the TeSSLa spec

1.2 Semantics of an ideal, synchronous evaluation Engine

An ideal implementation I for a specification T is one that consumes an input event e_x and immediately emits appropriate output Events $o_{1,1}, o_{1,2}, \ldots, o_{1,x}$ with $x \in \mathbb{N}_{\geq 0}$ and changes it's internal state S to save the fact, that the event e_x was received and optionally other properties generated by the evaluation of the spec that are needed by later computations.

Therefore it's behaviour for a single input can be described by two functions:

$$\Phi : S \times E \to S$$
$$\Theta : S \times E \to O^*$$

The behaviour for multiple inputs is the composition of the functions.

The semantics for the implementation I for a Stream of input events from E^* is given by the output Stream from O^* that is generated by the formulas.

1.3 Semantics of an asynchronous evaluation Engine

An asynchronous implementation A for a specification T has more complex characteristics: It's state is defined by the product of the States of it's nodes, where each node represents a primitive operation in the specification and the nodes are organized as a DAG.The output is specified by the concatination of outputs of the nodes that are marked as output nodes in the specification.

In contrast to I the asynchronous implementation takes multiple steps to produce an output from an input. During a step multiple things can happen at once:

- A new, external input Event can be consumed by a source in the DAG and is propagated to it's children
- Multiple internal Nodes, which have at least one new input buffered on their input queue can perform their computation and propagete their new output to their children.
- Multiple output nodes, which have at least one new input buffered on their input queue, can produce a new output.

The semantic of the implementation is given by the product of the semantics of it's nodes.

1.4 Equality of Semantics

A and S are considered equal if for a Series of input Events they produce the same output Events. A won't emit the outputs in the right order, but because each event holds the timestamp it was generated, they can be reordered so that the output will be exactly the same as the one from I

1.4.1 Equality of States

To proof the equality of both systems we have to proof the equality of the states A will take wile producing the outputs to the States I takes. Let $e_1, e_2, \ldots, e_x$ be the input events both implementations receive. The State of I will change every time a

new input ic received and will produce all outputs that could be computed. It can be visualized as:

Input:	e_1	e_2	e_3
new State:	I_1	I_2	I_3
new Outputs:	$(o_{1,1}, \ldots, o_{1,x})$	$(o_{2,1}, \ldots, o_{2,y})$	$(o_{3,1}, \ldots, o_{3,y})$

List of Figures

List of Tables

Colophon

This thesis was typeset with LaTeX 2$_\varepsilon$. It uses the *Clean Thesis* style developed by Ricardo Langner. The design of the *Clean Thesis* style is inspired by user guide documents from Apple Inc.

Download the *Clean Thesis* style at `http://cleanthesis.der-ric.de/`.

Declaration

You can put your declaration here, to declare that you have completed your work solely and only with the help of the references you mentioned.

Luebeck, November 20, 2016

Alexander Schramm