TESSLA—A Temporal Stream-based Specification Language

March 3, 2016

1 Introduction

Outline

1. purpose, motivation
   - online processing of data
   - monitoring of trace properties, specifically execution traces of programs
   - Functional reactive programming as related concept
2. What you describe with a TeSSLa specification
   - input, output streams
   - application of functions, composition of function
   -
3. Modelling data in terms of streams
   - timing model ($\mathbb{R}$,$\mathbb{N}$,$\mathbb{Q}$,...), restrictions to streams with discrete set of time stamps (event streams) or peace-wise constant streams (continuous stream)
   - continuous streams and event streams
4. Functions on streams and desired properties (in general)
   - small examples
   - causality, statefulness, time invariance
   - (composition lemmata)
5. TeSSLa syntax
   - base grammar with functions and type annotations
   - syntactical extensions: infix operators, named arguments, the "on"

6. Types
   - Generic types
   - Coercion
7. Functional semantics of operators, small examples
8. Larger example/case study
   - producer/consumer, ring buffer, ...

2 Syntax

This section describes the syntax of TeSSLa.

2.1 Basic Syntax

As basic syntactic elements of TeSSLa we consider a set of *types* T, *names* $\mathcal{N}$ and *function symbols* $\mathcal{F}$. Formally, the latter are two families $\mathcal{N} = (N_t)_{t \in T}$ and $\mathcal{F} = (F_{(t_1 \dots t_m)t_{m+1}})_{t_1 \dots t_{m+1} \in T^+}$ of distinct sets of symbols that correspond to a specific type or signature, respectively. We refer to the tuple $\Sigma = (T, \mathcal{N}, \mathcal{F})$ as a *signature* providing the basic syntactical elements of our specification language.

Built from those we define *terms* x of type $t \in T$ and sequences of terms seq by the grammar

$$\begin{aligned} x &::= n \mid f_c \mid f(seq) \mid x : t \\ seq &::= x \mid x, seq \end{aligned}$$

where $n \in N_t$, $f_c \in F_{()t}$ (i.e., constants) and $f \in F_{(t_1 \dots t_m)t}$.

2.2 Syntactical Extensions

We consider three syntactical extensions to the base syntax presented above. The first is an *on* operator, the second are *infix operators* and the third are *named arguments*.

Let us consider the *on* operator first. Therefore, for a set of function symbols $F_{(t_1 \dots t_m)t}$ let $F^s_{(t_1 \dots t_m)t} \subseteq F_{(t_1 \dots t_m)t}$ be the set of function symbols of stateless functions. Hence the *terms* x of a type $t \in T$ including the *on* operator can be defined by the grammar

$$\begin{aligned} x &::= n \mid f_c \mid f(seq) \mid \text{on}\, f_s(seq_s) \mid x : t \\ seq &::= x \mid x, seq \\ y &::= n \mid f_c \mid f_s(seq_s) \mid \text{on}\, f_s(seq_s) \mid y : t \\ seq_s &::= y \mid y, seq_s \end{aligned}$$

where $f_s \in F^s_{(t_1 \dots t_m)t}$ and the rest is defined as before.

TODO: !infix operators, named arguments!

3 Semantics

In this section, the semantics of TeSSLa is defined. All in all, the functions used in TeSSLa can be categorized in six different types. Before we start with the semantics of the function we will fix some notation first.

In the following we will use $\mathcal{C}$ and $\mathcal{E}$ for the sets of continuous and event streams, respectively. With $\mathcal{C}[T]$ we refer to the set of continuous streams that only contain elements of the type T (f.e. $\mathcal{C}[\mathbb{B}]$ is the set of continuous boolean streams). We will use $\mathbb{D}$ as the set of all values.

3.1 Arithmetic Functions

Arithmetic functions take a certain number of value streams of type integer and return a stream that contains either integer or boolean values. The semantics for the arithmetic functions are defined in the following.

The function $\text{add} : \mathcal{C}[\mathbb{N}] \times \mathcal{C}[\mathbb{N}] \to \mathcal{C}[\mathbb{N}]$ for adding the values of two integer streams is defined as follows:

$$\text{add}(n_1, n_2) = n_3 \text{ with } n_3(t) = n_1(t) + n_2(t)$$

The function $\text{sub} : \mathcal{C}[\mathbb{N}] \times \mathcal{C}[\mathbb{N}] \to \mathcal{C}[\mathbb{N}]$ for subtracting the values of the second integer stream from the value of the first integer stream is defined as follows:

$$\text{sub}(n_1, n_2) = n_3 \text{ with } n_3(t) = n_1(t) - n_2(t)$$

The function $\text{mul} : \mathcal{C}[\mathbb{N}] \times \mathcal{C}[\mathbb{N}] \to \mathcal{C}[\mathbb{N}]$ for multiplying the values of two integer streams is defined as follows:

$$\text{mul}(n_1, n_2) = n_3 \text{ with } n_3(t) = n_1(t) \cdot n_2(t)$$

The function $\text{geq} : \mathcal{C}[\mathbb{N}] \times \mathcal{C}[\mathbb{N}] \to \mathcal{C}[\mathbb{B}]$ for checking if the values of the first integer stream is at least as big as the value of the second integer stream is defined as follows:

$$\text{geq}(n_1, n_2) = b_1 \text{ with } b_1(t) = n_1(t) \geq n_2(t)$$

The function $\text{leq} : \mathcal{C}[\mathbb{N}] \times \mathcal{C}[\mathbb{N}] \to \mathcal{C}[\mathbb{B}]$ for checking if the values of the first integer stream is at least as big as the value of the second integer stream is defined as follows:

$$\text{leq}(n_1, n_2) = b_1 \text{ with } b_1(t) = n_1(t) \leq n_2(t)$$

The function $\text{equals} : \mathcal{C}[\mathbb{N}] \times \mathcal{C}[\mathbb{N}] \to \mathcal{C}[\mathbb{B}]$ for checking if the values of two integer streams are equal is defined as follows:

$$\text{equals}(n_1, n_2) = b_1 \text{ with } b_1(t) = \begin{cases} \top & \text{if } n_1(t) = n_2(t) \\ \bot & \text{else} \end{cases}$$

The function max : $\mathcal{C}[\mathbb{N}] \times \mathcal{C}[\mathbb{N}] \to \mathcal{C}[\mathbb{N}]$ for getting the maximum of the current values of two integer streams is defined as follows:

$$\max(n_1, n_2) = n_3 \text{ with } n_3(t) = \begin{cases} n_1(t) & \text{if } n_1(t) > n_2(t) \\ n_2(t) & \text{else} \end{cases}$$

The function min : $\mathcal{C}[\mathbb{N}] \times \mathcal{C}[\mathbb{N}] \to \mathcal{C}[\mathbb{N}]$ for getting the minimum of the current values of two integer streams is defined as follows:

$$\max(n_1, n_2) = n_3 \text{ with } n_3(t) = \begin{cases} n_1(t) & \text{if } n_1(t) < n_2(t) \\ n_2(t) & \text{else} \end{cases}$$

The function not : $\mathcal{C}[\mathbb{B}] \to \mathcal{C}[\mathbb{B}]$ for getting the negation of the value of a boolean stream is defined as follows:

$$\text{not}(b_1) = b_2 \text{ with } b_2(t) = \neg b_1(t)$$

The function or : $\mathcal{C}[\mathbb{B}] \times \mathcal{C}[\mathbb{B}] \to \mathcal{C}[\mathbb{B}]$ for combining two boolean streams with an or is defined as follows:

$$\text{or}(b_1, b_2) = b_3 \text{ with } b_3(t) = b_1(t) \vee b_2(t)$$

3.2 Timing Functions

Timing functions take a certain number of continuous streams and return a continuous or event stream. The semantics for the timing functions are defined in the following.

The function delay : $\mathcal{C} \times \mathbb{Q} \to \mathcal{C}$ for delaying the values of the input stream for a certain amount of time is defined as follows:

$$\text{delay}(c_1, q) = c_2 \text{ with } c_2(t) = c_1(t - q)$$

The function getTimestamp : $\mathcal{C} \to \mathbb{Q}$ for replacing the values of the input stream with its timestamps is defined as follows:

$$\text{getTimestamp}(c_1) = c_2 \text{ with } c_2(t) = t$$

The function inPast : $\mathcal{C} \times \mathbb{D} \times \mathbb{Q} \to \mathcal{C}[\mathbb{B}]$ for checking if a certain value holds in a given time frame in the past is defined as follows:

$$\text{inPast}(c_1, d, q) = b \text{ with } b(t) = \begin{cases} \top & \text{if } \exists t - q \leq t' \leq t : c_1(t') = d \\ \bot & \text{else} \end{cases}$$

The function synchronize : $\mathcal{C} \times \mathcal{C} \times \mathbb{Q} \to \mathcal{E}$ for checking whether a corresponding event occurs in the second stream for each event in the first stream within a certain time frame is defined as follows:

$$\text{synchronize}(c_1, c_2, q) = e_1 \text{ with } ???$$

3.3 Aggregations

3.4 Selectors/Filters/Conditionals/Combinators

3.5 On

3.6 Monitors