

Streaming Functions in riff

Glyn Normington

January 17, 2019

This document provides a formal model of streaming functions in the riff FaaS project (<https://projectriff.io>).

Contents

1	Introduction	1
2	Overview of this document	1
3	Fundamentals	1
4	Reactive Streams	2
5	Streaming Functions	2
A	Z Notation	3

1 Introduction

This document provides a formal model of streaming functions in the riff project.

2 Overview of this document

Event streaming is fraught with ambiguous terminology. For example, the word stream has multiple, overlapping connotations. This document attempts to model streaming functions precisely. The model is abstract in the sense that it does not capture all facets of streaming functions, but those it does capture, it captures precisely and unambiguously.

The Z specification language is used to capture the model, but sufficient English text is also provided that readers who do not know Z should be able to understand the model. The appendix contains a summary of the Z notation. For more information about Z, please consult the Z Manual (<https://www.cs.umd.edu/~mvz/handouts/z-manual.pdf>). The model was type checked using fuzz (<https://bitbucket.org/Spivey/fuzz>).

3 Fundamentals

Streams consist of messages, but for our current purposes, the details of a message are irrelevant. Similarly, errors are modelled but their details are not.

$[Message, Error]$

4 Reactive Streams

A reactive stream consist of an ordered series of zero or more messages optionally followed by a completion signal or an error.

$$RStream ::= Next\langle\langle Message \times RStream \rangle\rangle \mid \\ Complete \mid \\ Failed\langle\langle Error \rangle\rangle$$

5 Streaming Functions

A streaming function takes a reactive stream as input and produces another reactive stream as output.

$$StreamingFun == RStream \rightarrow RStream$$

Streaming functions may be chained together to produce another streaming function. The input stream of the chain is input to the first function in the chain. The output stream of the first function is the input stream of the second function and so forth. The output stream of the chain is the output stream of the last function in the chain.

$$\begin{array}{|l} Chain : seq_1 StreamingFun \rightarrow StreamingFun \\ \hline \forall f : StreamingFun; fs : seq_1 StreamingFun \bullet \\ Chain(\langle f \rangle) = f \wedge \\ Chain(\langle f \rangle \frown fs) = Chain(fs) \circ f \end{array}$$

To do: introduce channels, e.g. between chained functions.

A Z Notation

Numbers:

\mathbb{N} Natural numbers $\{0, 1, \dots\}$

Propositional logic and the schema calculus:

$\dots \wedge \dots$	And	$\langle\langle \dots \rangle\rangle$	Free type injection
$\dots \vee \dots$	Or	$[\dots]$	Given sets
$\dots \Rightarrow \dots$	Implies	$', ?, !, 0 \dots 9$	Schema decorations
$\forall \dots \mid \dots \bullet \dots$	For all	$\dots \vdash \dots$	theorem
$\exists \dots \mid \dots \bullet \dots$	There exists	$\theta \dots$	Binding formation
$\dots \setminus \dots$	Hiding	$\lambda \dots$	Function definition
$\dots \hat{=} \dots$	Schema definition	$\mu \dots$	Mu-expression
$\dots == \dots$	Abbreviation	$\Delta \dots$	State change
$\dots ::= \dots \mid \dots$	Free type definition	$\Xi \dots$	Invariant state change

Sets and sequences:

$\{\dots\}$	Set	$\dots \setminus \dots$	Set difference
$\{\dots \mid \dots \bullet \dots\}$	Set comprehension	$\bigcup \dots$	Distributed union
$\mathbb{P} \dots$	Set of subsets of	$\# \dots$	Cardinality
\emptyset	Empty set	$\dots \subseteq \dots$	Subset
$\dots \times \dots$	Cartesian product	$\dots \subset \dots$	Proper subset
$\dots \in \dots$	Set membership	$\dots \text{ partition } \dots$	Set partition
$\dots \notin \dots$	Set non-membership	seq	Sequences
$\dots \cup \dots$	Union	$\langle \dots \rangle$	Sequence
$\dots \cap \dots$	Intersection	disjoint \dots	Disjoint sequence of sets

Functions and relations:

$\dots \leftrightarrow \dots$	Relation	\dots^*	Reflexive-transitive closure
$\dots \rightarrow \dots$	Partial function	$\dots (\dots)$	Relational image
$\dots \rightarrow \dots$	Total function	$\dots \oplus \dots$	Functional overriding
$\dots \mapsto \dots$	Partial injection	$\dots \triangleleft \dots$	Domain restriction
$\dots \mapsto \dots$	Injection	$\dots \triangleright \dots$	Range restriction
dom \dots	Domain	$\dots \trianglelefteq \dots$	Domain subtraction
ran \dots	Range	$\dots \trianglerighteq \dots$	Range subtraction
$\dots \mapsto \dots$	maplet		
$\dots \sim \dots$	Relational inverse		

Axiomatic descriptions:

<i>Declarations</i>
<i>Predicates</i>

Schema definitions:

<i>SchemaName</i>
<i>Declaration</i>
<i>Predicates</i>