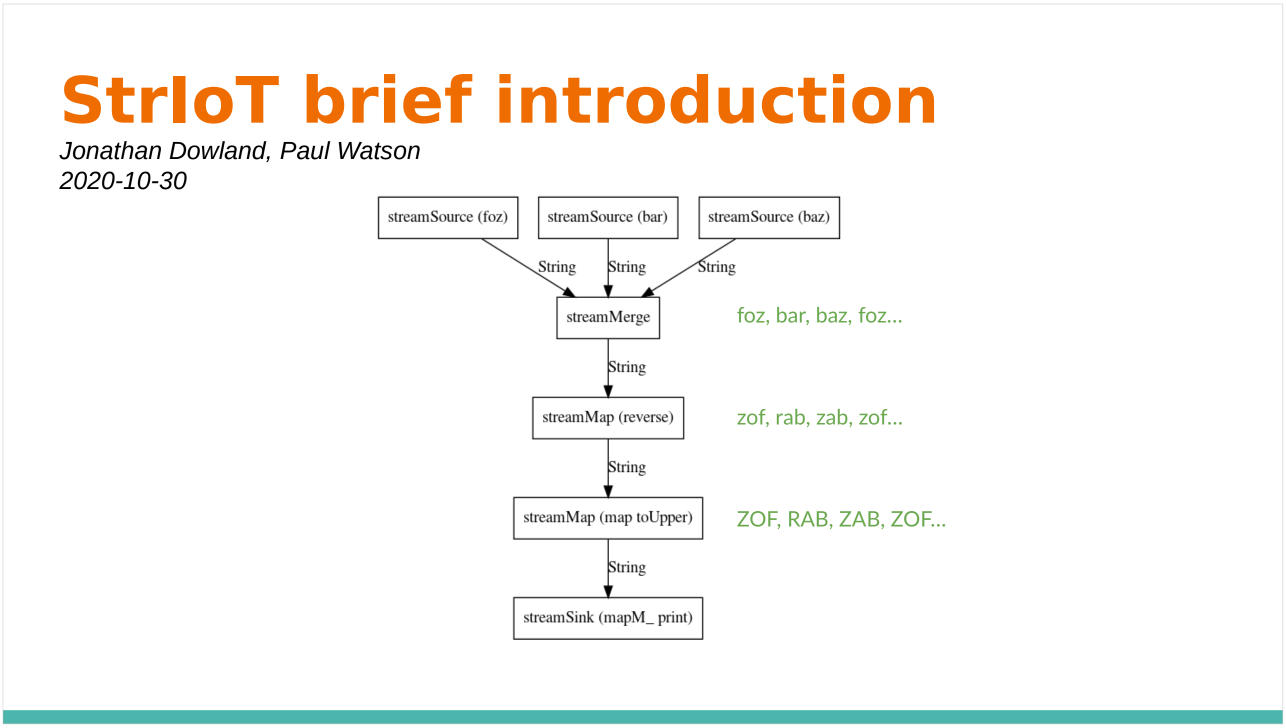
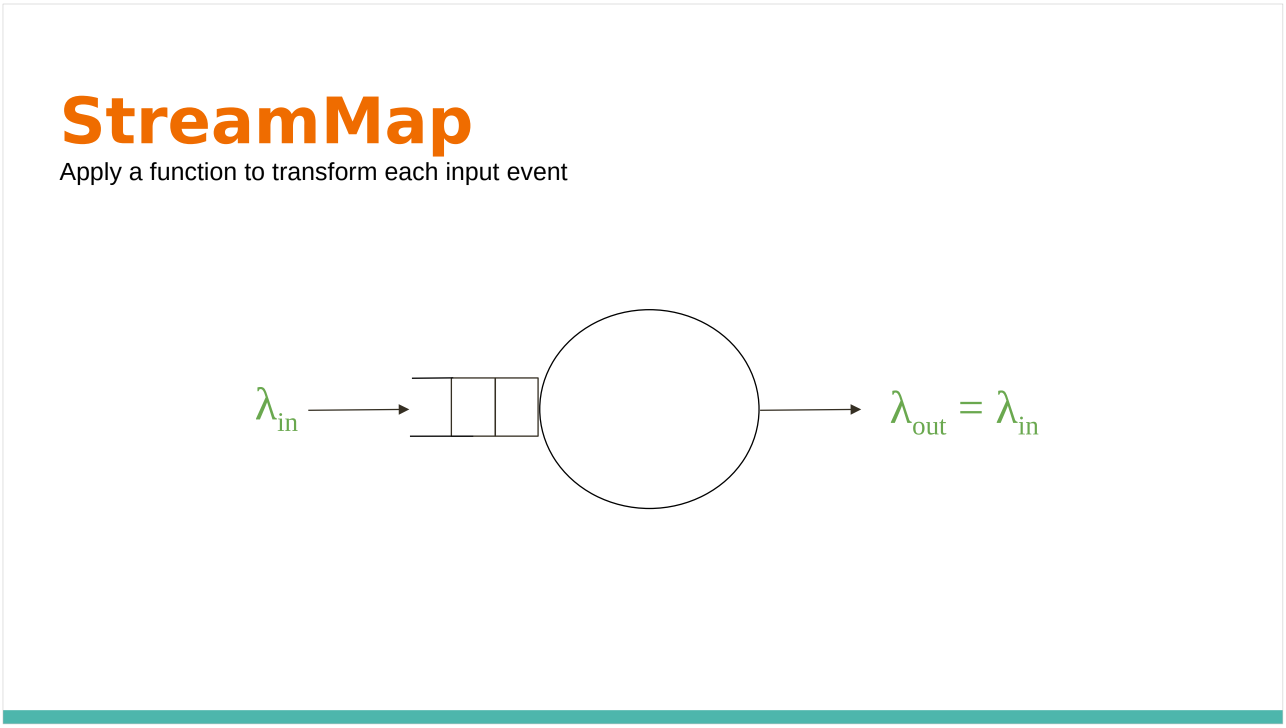


StrIoT brief introduction

Jonathan Dowland, Paul Watson
2020-10-30

streamSource (foz)
streamSource (bar)
streamSource (baz)
String
String
String
streamMerge
foz, bar, baz, foz…
String
streamMap (reverse)
zof, rab, zab, zof…
String
streamMap (map toUpper)
ZOF, RAB, ZAB, ZOF…
String
streamSink (mapM_ print)

User composes a stream-processing program in terms of “operators”: result is a (reverse) tree, with ≥1 source nodes and rooted in a single sink node.



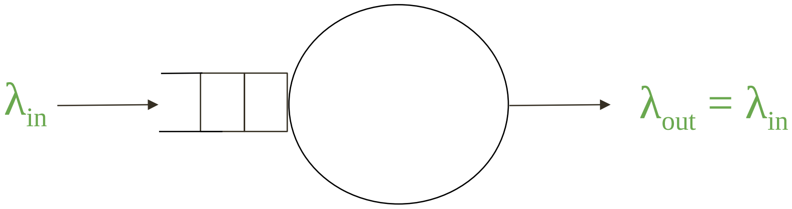
StreamMap

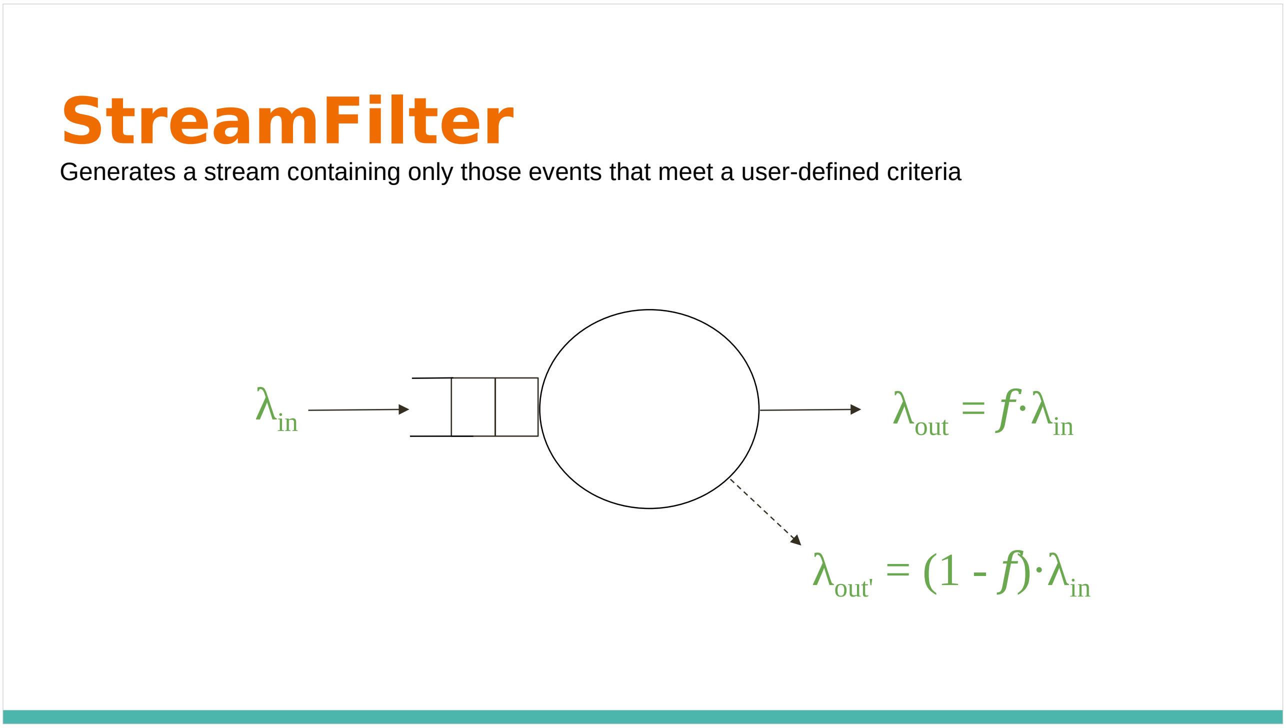
Apply a function to transform each input event

We’ll go through the 8 operators brieflyand show how they may be represented in a Jackson network model
The operator description are taken from the README at striot.org

StreamScan

Apply a function to transform each input event, taking into account past history





StreamFilter

Generates a stream containing only those events that meet a user-defined criteria

λ_{in}

$\lambda_{out} = f \cdot \lambda_{in}$

$\lambda_{out'} = (1 - f) \cdot \lambda_{in}$

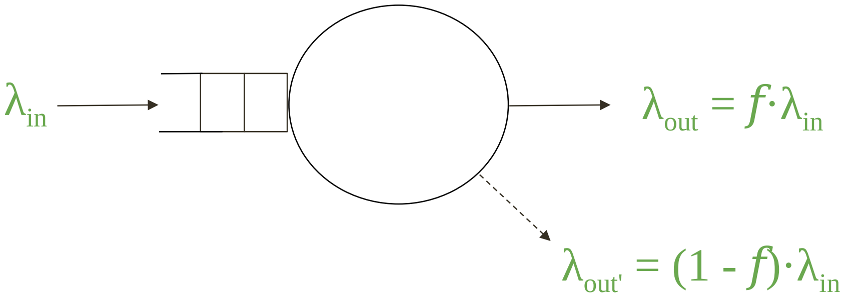
f is the selectivity of the operator

λout' is the rate of the discarded events

To balance the network equations we've introduced a "phantom" output

StreamFilterAcc

Filter events, taking into account past results



λ_{in}

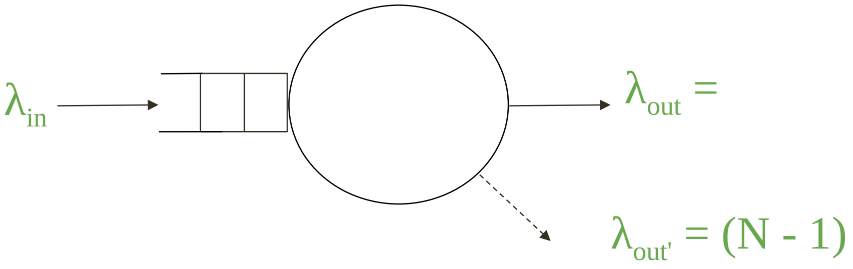
$\lambda_{out} = f \cdot \lambda_{in}$

$\lambda_{out'} = (1 - f) \cdot \lambda_{in}$

f is the selectivity of the operator
λout' is the rate of the discarded events

StreamWindow chop

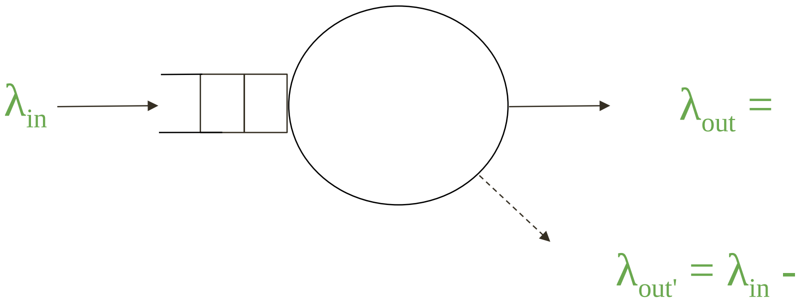
Create non-overlapping windows that are a fixed number (N) events in length



The modelling of streamWindow depends upon the user-supplied "window maker" parameter. Here we discuss four example window makers

StreamWindow chopTime

Create non-overlapping windows that are a fixed time (T) length



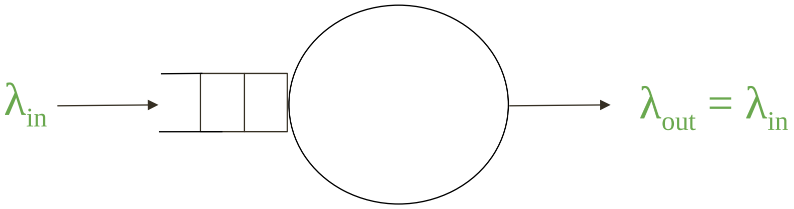
λ_{in}

$\lambda_{out} =$

$\lambda_{out'} = \lambda_{in}$ -

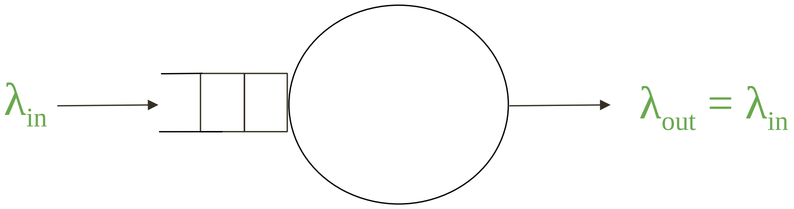
StreamWindow sliding

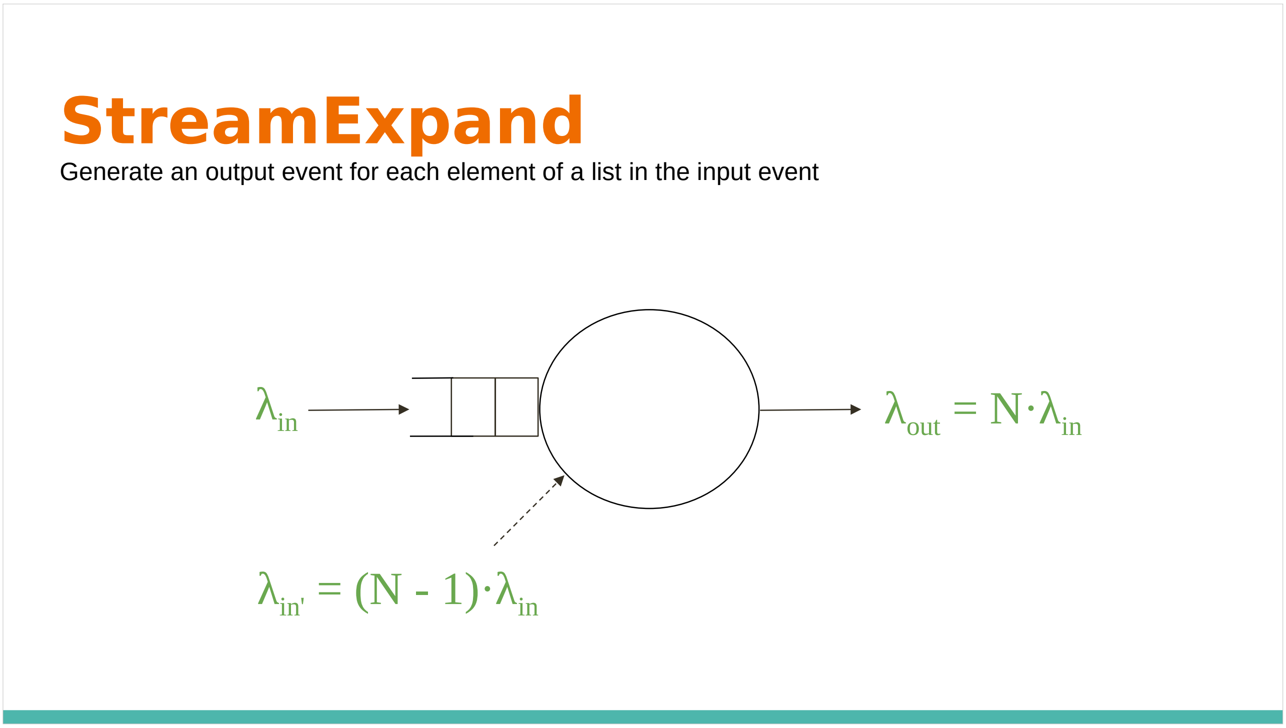
Create overlapping windows that are a fixed number of events in length



StreamWindow slidingTime

Create overlapping windows that are a fixed time length





StreamExpand

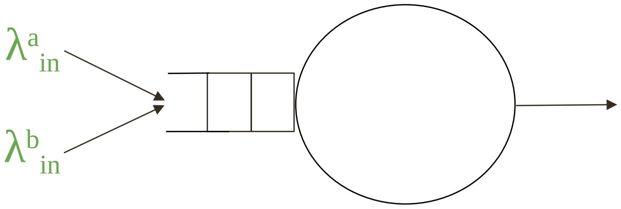
Generate an output event for each element of a list in the input event

λ_{in}
$\lambda_{out} = N\cdot\lambda_{in}$
$\lambda_{in'} = (N - 1)\cdot\lambda_{in}$

The dual of windowing
"N" is the average number of elements in a list
Here to balance the network equations we've introduced a "phantom" input

StreamMerge

Combine ≥2 streams by merging their events

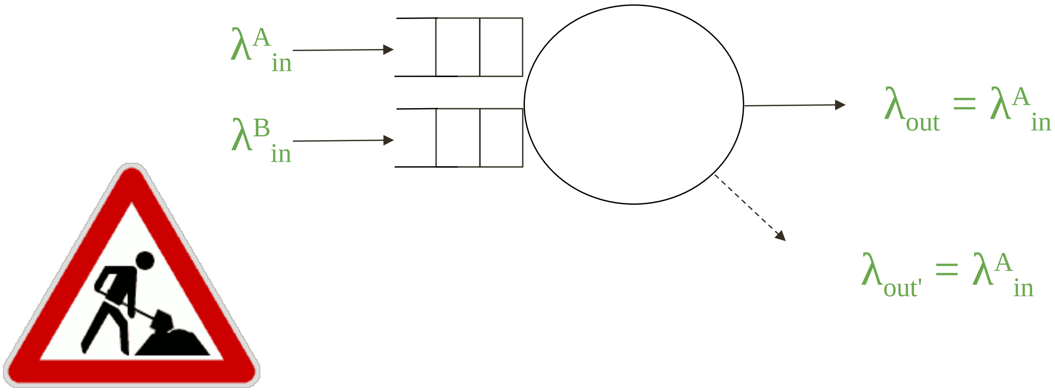


$$\lambda_{out} = \lambda^a{}_{in} + \lambda^b{}_{in}$$

Here representing two inputs but could be more

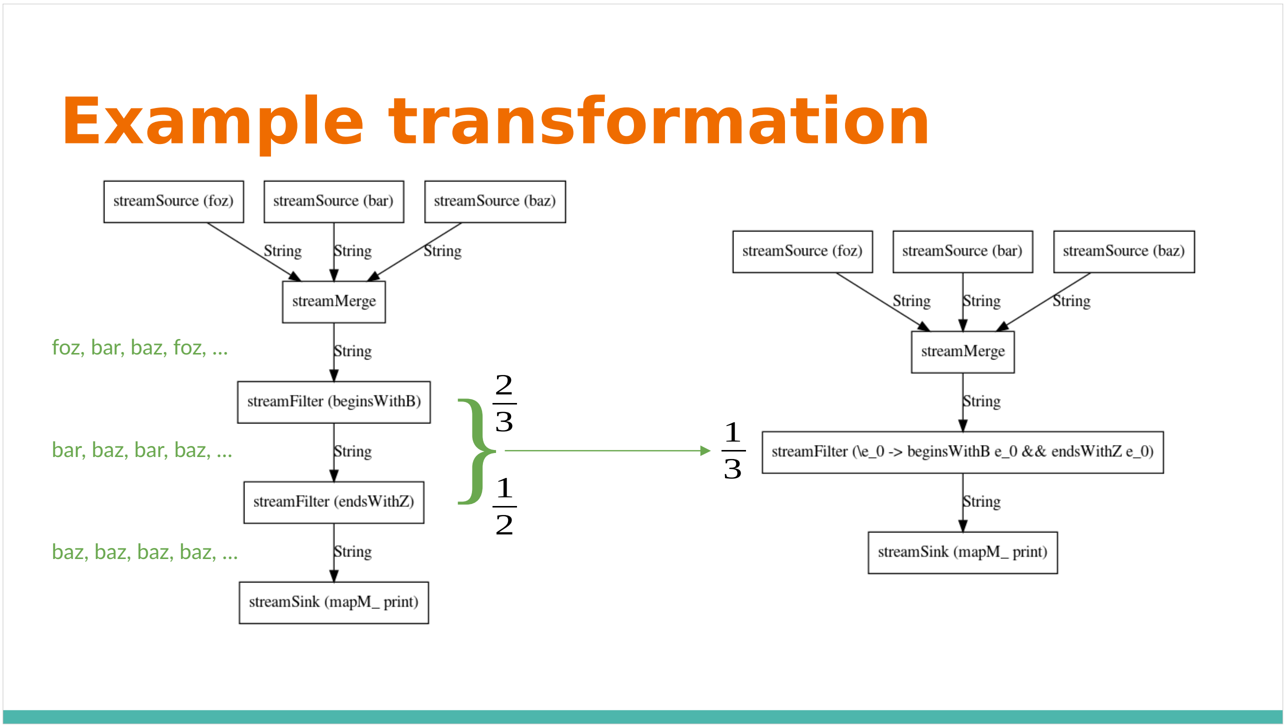
StreamJoin

Combine two streams by creating a tuple for each pair of input events


λ^{A}_{in}
λ^{B}_{in}
$\lambda_{out} = \lambda^{A}_{in}$
$\lambda_{out'} = \lambda^{A}_{in}$

This one is a work in progress

λAin = λBin



Example transformation

streamSource (foz)

streamSource (bar)

streamSource (baz)

String

String

String

streamMerge

foz, bar, baz, foz, …

String

streamFilter (beginsWithB)

$\frac{2}{3}$

bar, baz, bar, baz, …

String

streamFilter (endsWithZ)

$\frac{1}{2}$

baz, baz, baz, baz, …

String

streamSink (mapM_ print)

streamSource (foz)

streamSource (bar)

streamSource (baz)

String

String

String

streamMerge

String

$\frac{1}{3}$

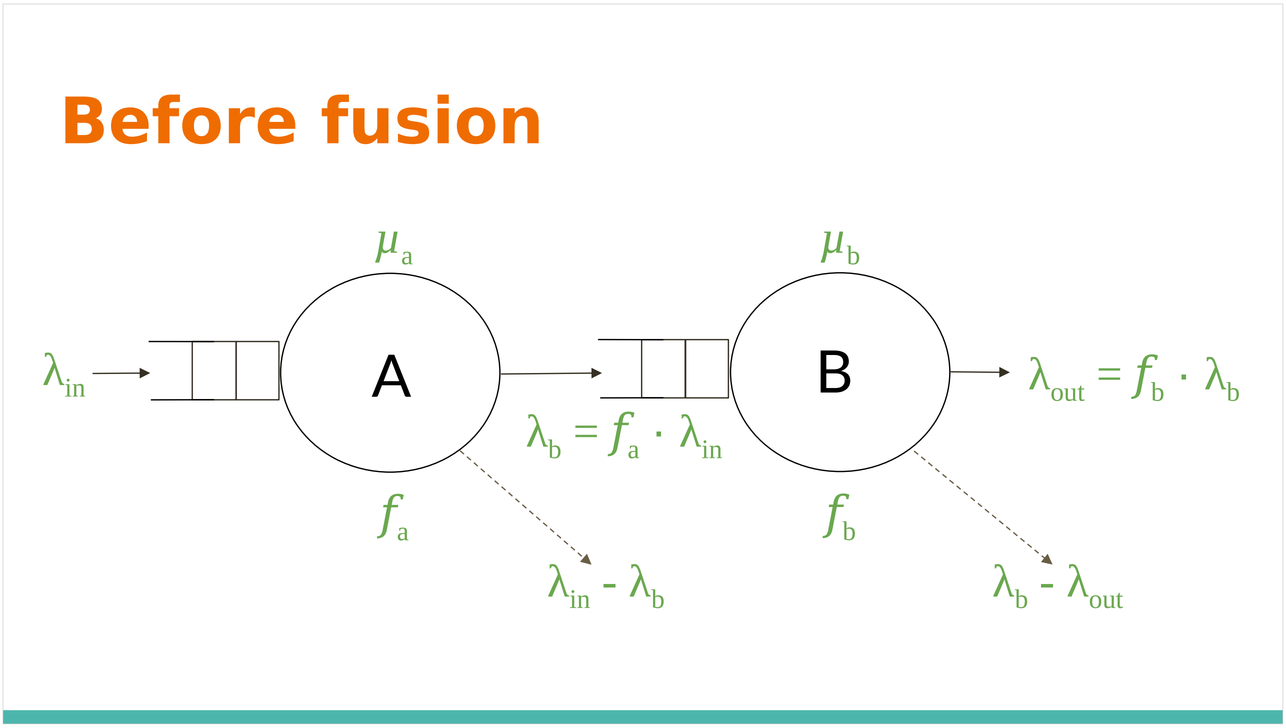
streamFilter (\e_0 -> beginsWithB e_0 && endsWithZ e_0)

String

streamSink (mapM_ print)

The design of Striot means we are able to rewrite a stream-processing program: rewrites preserve the functional behaviour but may change other aspects (e.g. performance)

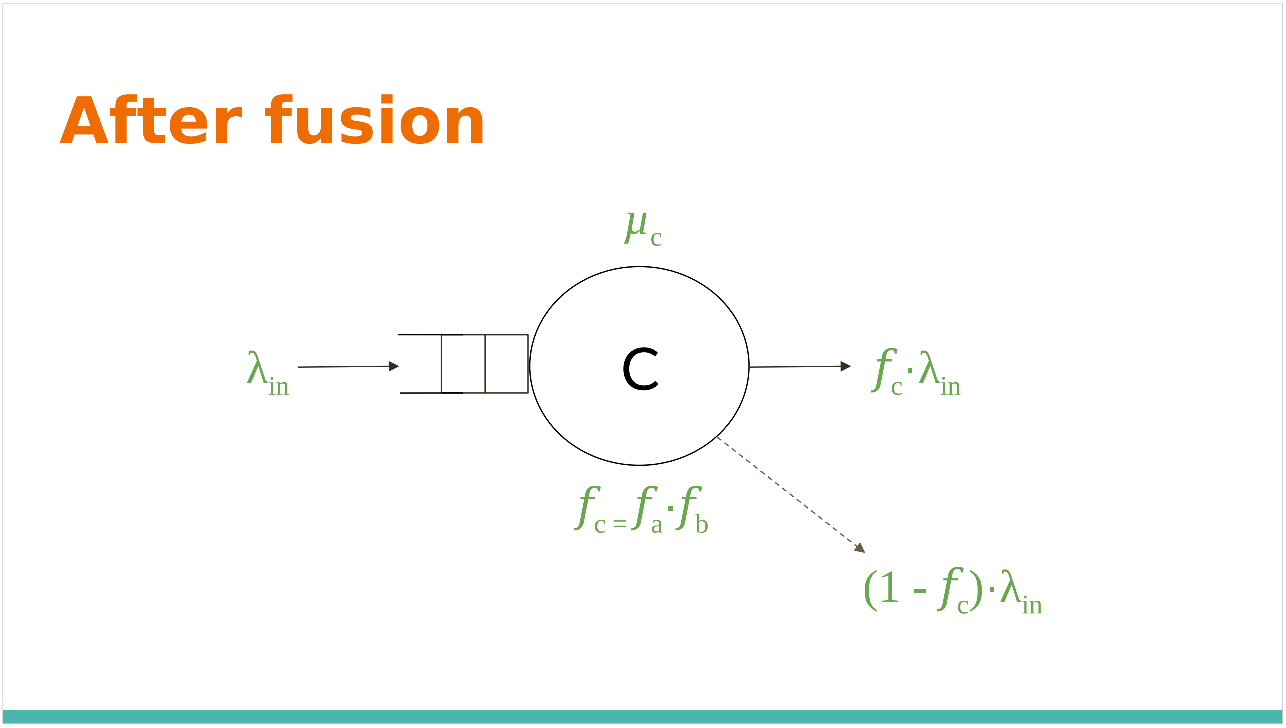
Here we have replaced two serially-arranged “filter” nodes with a single “filter” which performs the work of both of the original operators.



Before fusion

μ_a
μ_b
λ_{in}
A
B
$\lambda_{out} = f_b \cdot \lambda_b$
$\lambda_b = f_a \cdot \lambda_{in}$
f_a
f_b
$\lambda_{in} - \lambda_b$
$\lambda_b - \lambda_{out}$

This slide is busier than I'd like! The input and output arrival rates and the selectivity of each operator are known

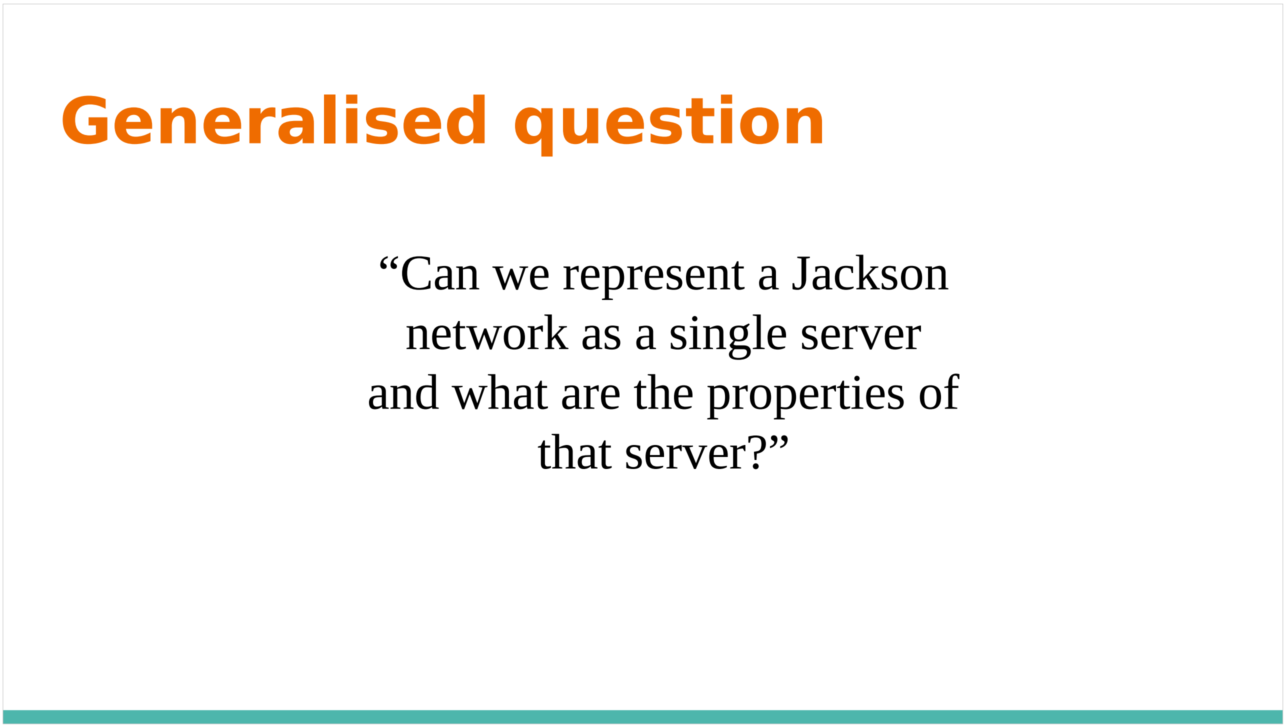


After fusion

What are the properties of the new Server, performing the work of the two older Servers?
We are confident that we know the output rates and the filter selectivity: the product of the input selectivities

But what of the mean service rate?

Intuitively Since the new server is doing the work of the two original servers, it seems the mean average service time should be the sum of the mean average service times for the two servers. This does not model any efficiency gains (or losses)



Generalised question

"Can we represent a Jackson network as a single server and what are the properties of that server?"

I've seen the technique of applying Little's Theorem to a sub-part of a Jackson network, but Little's Theorem does not appear to help us determine the mean service rate because it is not defined in terms of service rates