SHACC

A functional animator for a component calculus

André Martins, Luís S. Barbosa and Nuno F.Rodrigues

SHACC - tool

- SHACC is a component-based system
  - Able define and animate systems expressed in a component calculus

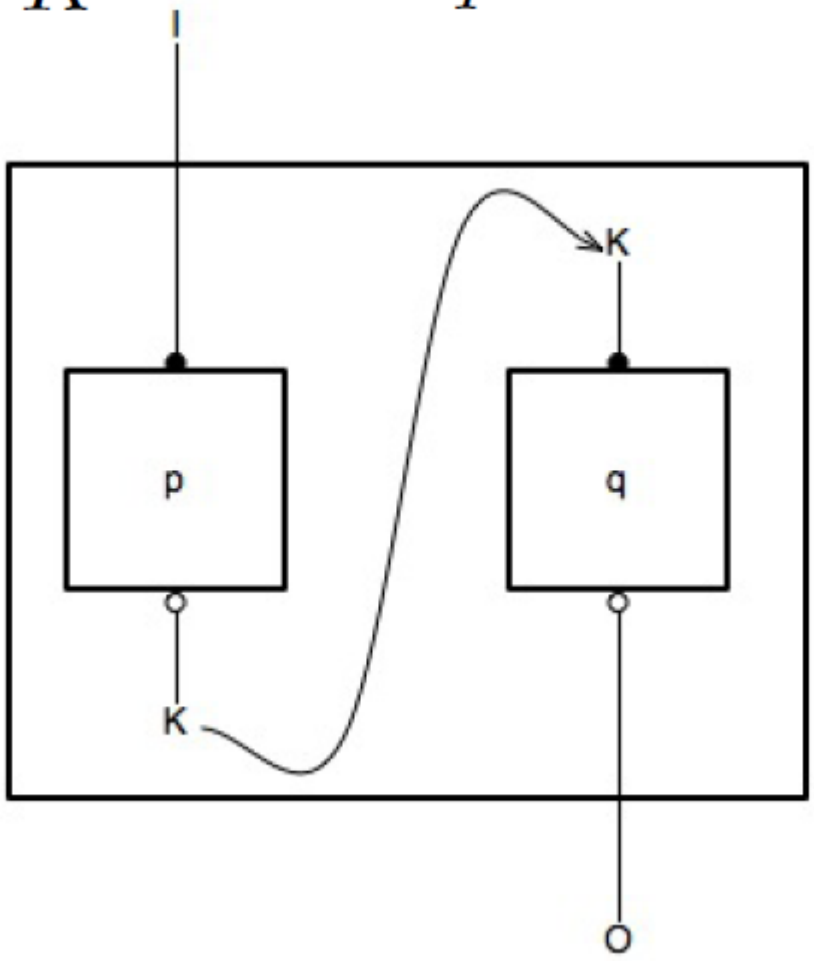
A components' calculus

$$\langle u_p \in U_p, \overline{a}_p : U_p \longrightarrow \mathsf{B}(U_p \times O)^I \rangle$$

Where:

- up: is the initial state
- B: is a strong monad capturing the component behavior model
- I , O is a collection of sets acting as component interfaces

$p : I \longrightarrow K$ $\qquad$ $q : K \longrightarrow O$



$$p\,;q \;=\; \langle \langle u_p, u_q \rangle \in U_p \times U_q, \overline{a}_{p;q} \rangle$$

Where:

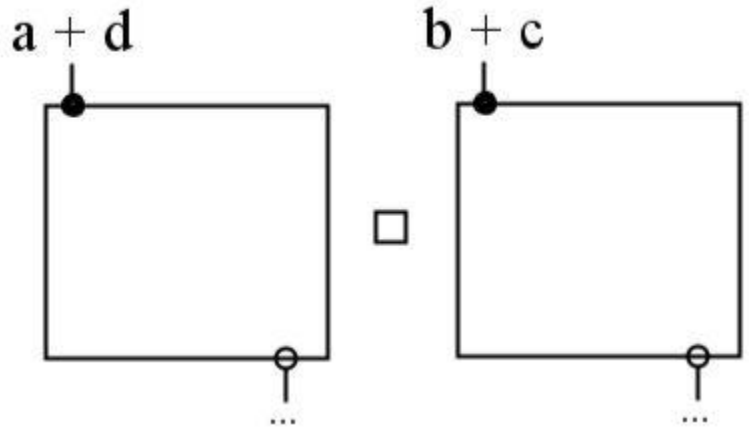
$$\begin{aligned} a_{p;q} = {} & U_p \times U_q \times I \xrightarrow{\times r} U_p \times I \times U_q \xrightarrow{a_p \times \mathsf{id}} \\ & \mathsf{B}(U_p \times K) \times U_q \xrightarrow{\tau_r} \mathsf{B}(U_p \times K \times U_q) \xrightarrow{\mathsf{B}(a \cdot \times r)} \\ & \mathsf{B}(U_p \times (U_q \times K)) \xrightarrow{\mathsf{B}(\mathsf{id} \times a_q)} \mathsf{B}(U_p \times \mathsf{B}(U_q \times O)) \\ & \xrightarrow{\mathsf{B}\tau_l} \mathsf{BB}(U_p \times (U_q \times O)) \xrightarrow{\mathsf{BB}a^\circ} \\ & \mathsf{BB}(U_p \times U_q \times O) \xrightarrow{\mu} \mathsf{B}(U_p \times U_q \times O) \end{aligned}$$

Using Haskell to describing the calculus

```
seqCompostion :: Strong m =>
((u,i)-> m (u,k)) ->
((v,k)-> m (v,o)) ->
((u,v), i) -> m ((u,v),o)

seqCompostion p q = mult . (fmap (fmap assocl)). (fmap lstr).
                    (fmap (id >< q)) . (fmap xl).
                    rstr . (p >< id) . xr
```

Extension of the prototyper with a behavioural customization

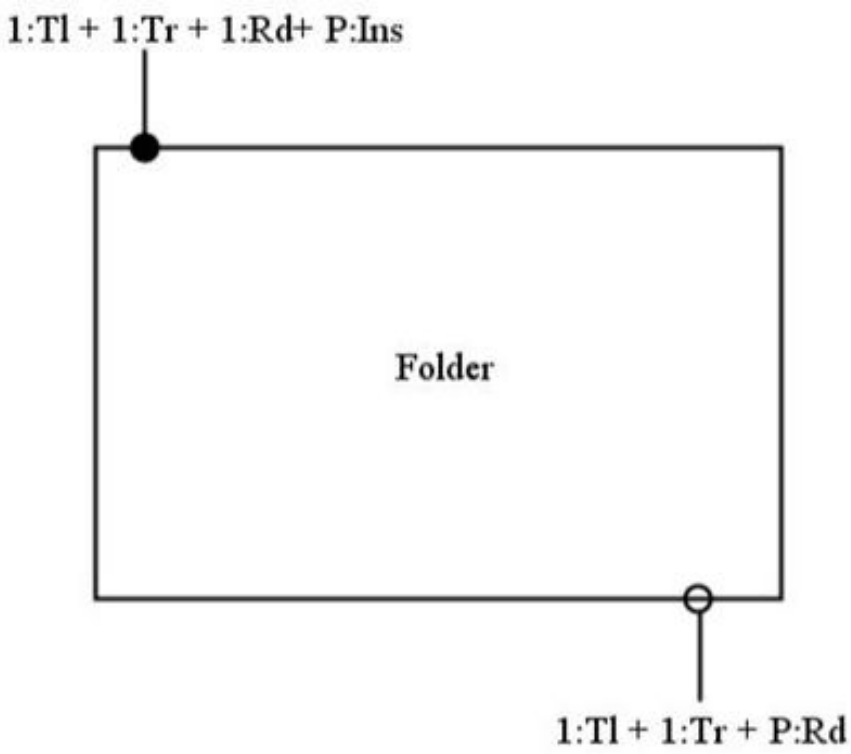
(a.b | a.c| d)* + 

=

Safe mode

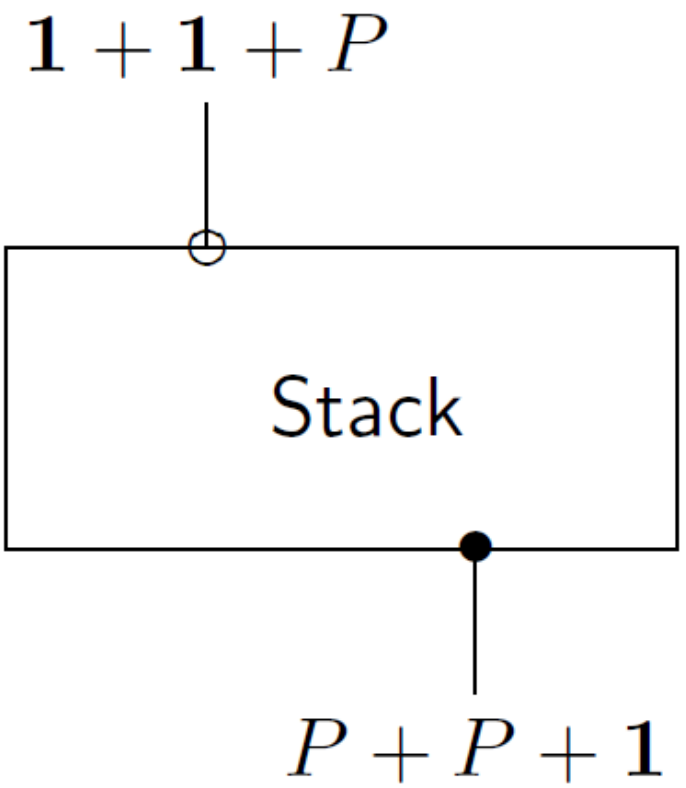
Example: Folder

1. Turn page left – **Tl**
2. Turn page right – **Tr**
3. Read- **Rd**
4. Insert – **Ins**



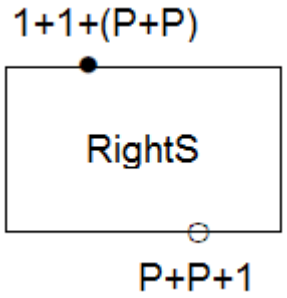
Stack

$$\begin{cases} \text{pop}: & \mathbf{1} \longrightarrow P \\ \text{top}: & \mathbf{1} \longrightarrow P \\ \text{push}: & P \longrightarrow \mathbf{1} \end{cases}$$

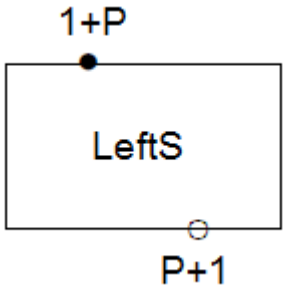


Defining Components

$$\mathrm{RightS} = \mathrm{Stack}[\mathrm{id} + \bigtriangledown, id] : 1 + 1 + (P + P) \rightarrow P + P + 1$$



$$\mathrm{LeftS} = \mathrm{Stack}[\mathrm{i}_2 + Id, (id + !_{p+1}) \bullet a_+] : 1 + P \rightarrow P + 1$$

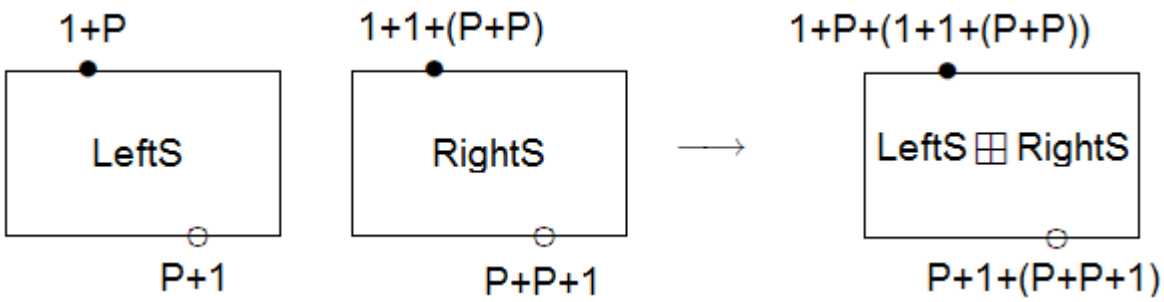


Choice ($\boxplus$)

$$\text{RightS} = \text{Stack}[\text{id} + \bigtriangledown, id] : 1 + 1 + (P + P) \rightarrow P + P + 1$$

$$\text{LeftS} = \text{Stack}[\text{i}_2 + Id, (id + !_{p+1}) \bullet a_+] : 1 + P \rightarrow P + 1$$

$$LeftS \boxplus \text{RightS} : 1+\text{P}+(1+1+(\text{P}+\text{P})) \rightarrow \text{P}+1+(\text{P}+\text{P}+1)$$



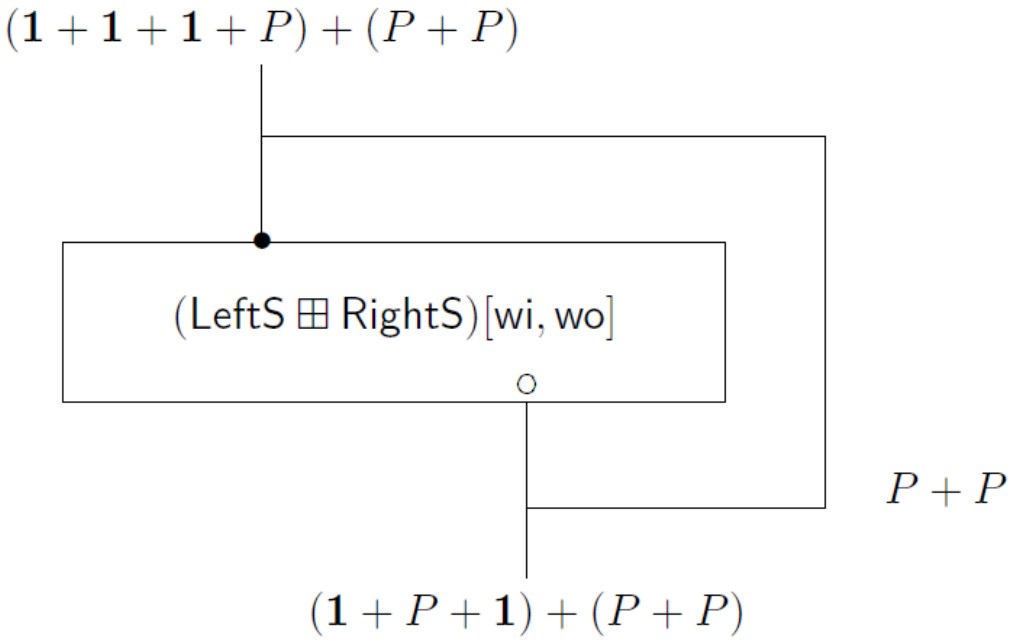
Combinator Hook (↰)

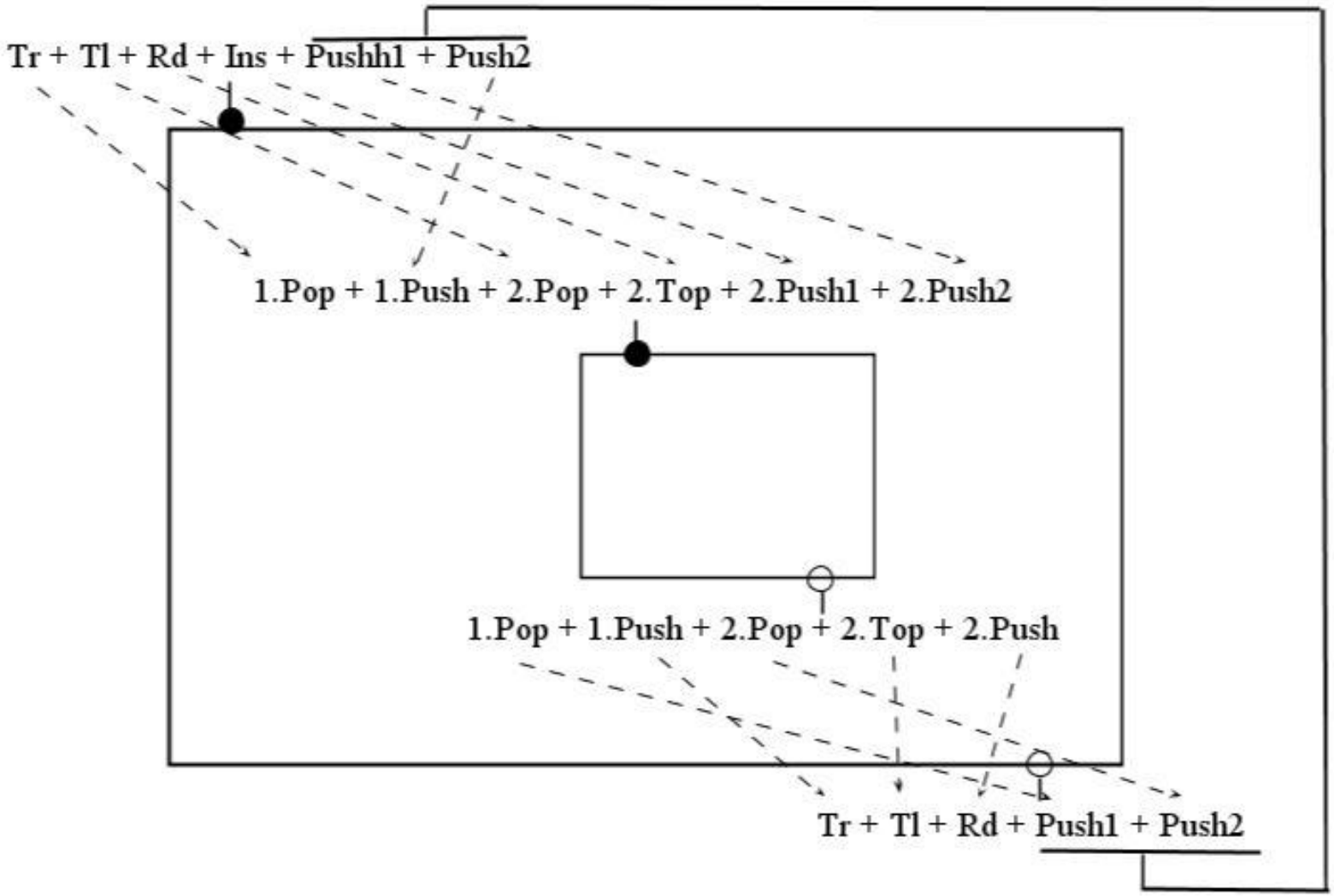
$$\text{AlmostFolder} = ((\text{LeftS} \boxplus RightS)[wi, wo]) \; \hookleftarrow_{P+P}$$

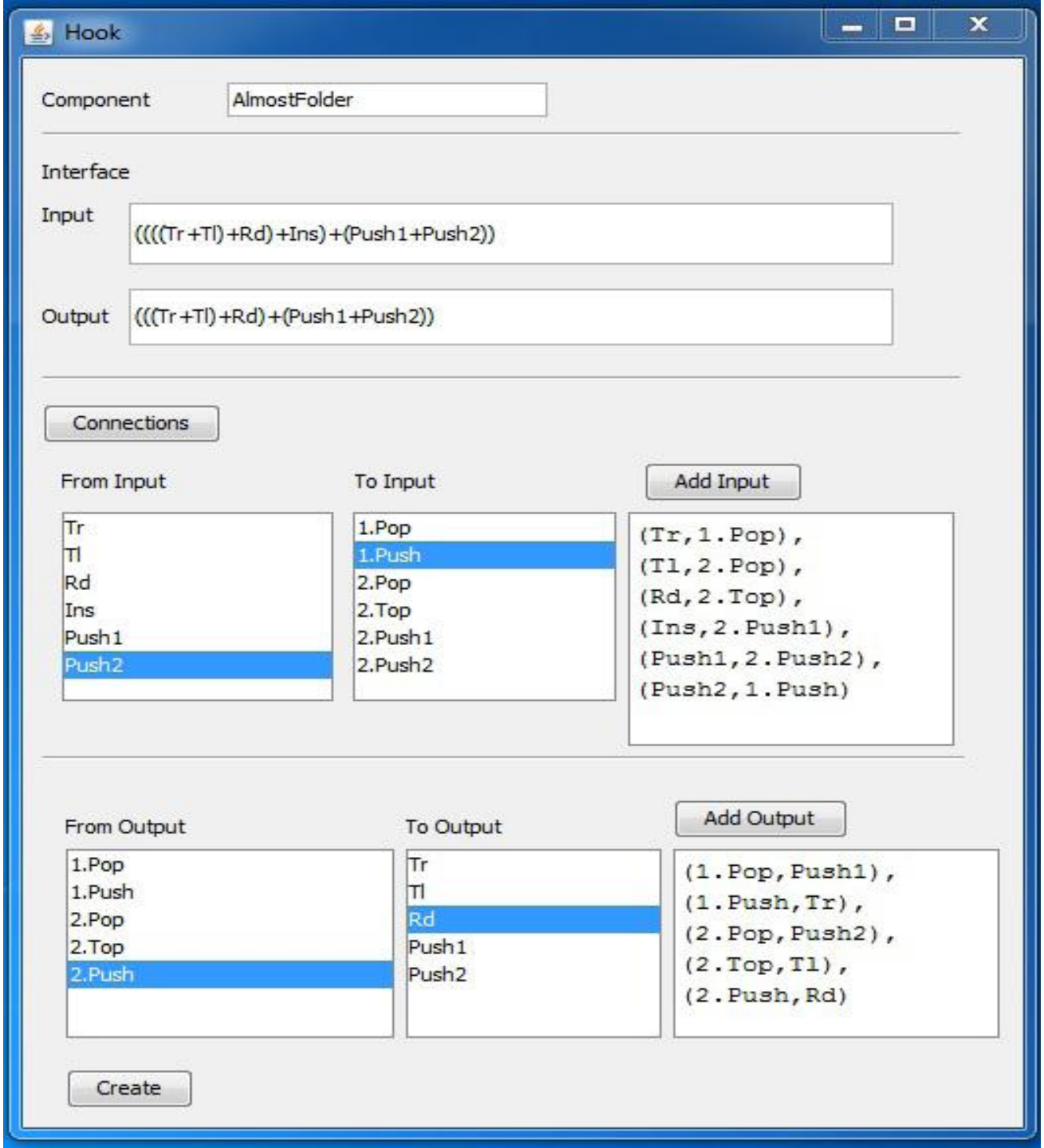
Where,

$wi = [[[[i_{11}, i_{211}], i_{212}], i_{222}], [i_{221}, i_{12}]]$

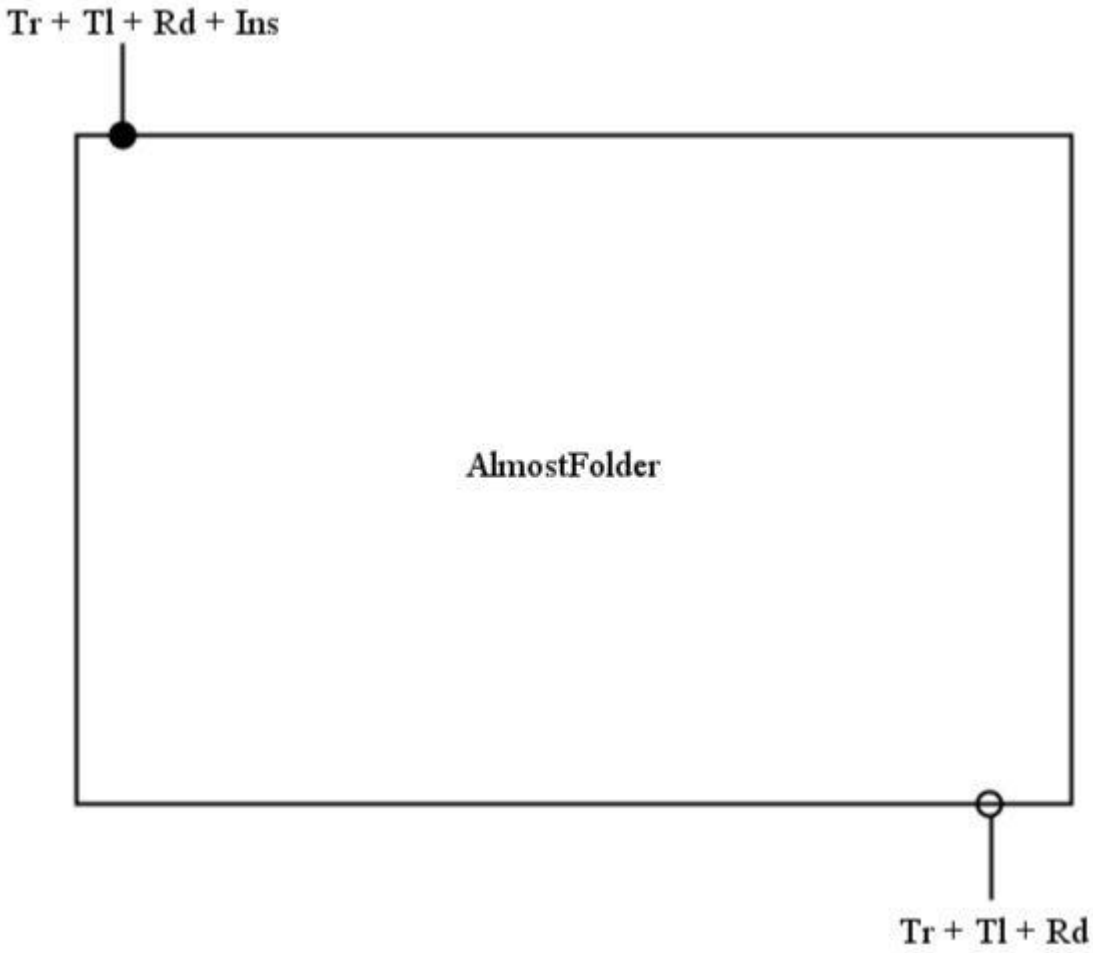
$wo = [[i_{21}, i_{111}], [[i_{22}, i_{112}], i_{12}]]$




Tr + Tl + Rd + Ins + Pushh1 + Push2
1.Pop + 1.Push + 2.Pop + 2.Top + 2.Push1 + 2.Push2
1.Pop + 1.Push + 2.Pop + 2.Top + 2.Push
Tr + Tl + Rd + Push1 + Push2


Hook
Component
AlmostFolder
Interface
Input
((((Tr+Tl)+Rd)+Ins)+(Push1+Push2))
Output
(((Tr+Tl)+Rd)+(Push1+Push2))
Connections
From Input
To Input
Add Input
Tr
Tl
Rd
Ins
Push1
Push2
1.Pop
1.Push
2.Pop
2.Top
2.Push1
2.Push2
(Tr,1.Pop),
(Tl,2.Pop),
(Rd,2.Top),
(Ins,2.Push1),
(Push1,2.Push2),
(Push2,1.Push)
From Output
To Output
Add Output
1.Pop
1.Push
2.Pop
2.Top
2.Push
Tr
Tl
Rd
Push1
Push2
(1.Pop,Push1),
(1.Push,Tr),
(2.Pop,Push2),
(2.Top,Tl),
(2.Push,Rd)
Create

Providing the final interface of Folder



Demo…

Future work!

- Adding Quality of Service to calculus
  - The calculus to include the QoS
- SHACC
  - Implementation of calculus with QoS
  - Improving User-Interface

Questions?