

# Memoryful Geometry of Interaction II

Recursion and Adequacy

Koko Muroya  
(Univ. Tokyo)

Naohiko Hoshino  
(Kyoto Univ.)

Ichiro Hasuo  
(Univ. Tokyo)

# Geometry of Interaction (GoI)

semantics of  $\left\{ \begin{array}{l} \text{linear logic proofs [Girard '89]} \\ \text{functional programs} \end{array} \right.$

token machine semantics

[Mackie '95] [Danos & Renier '99]

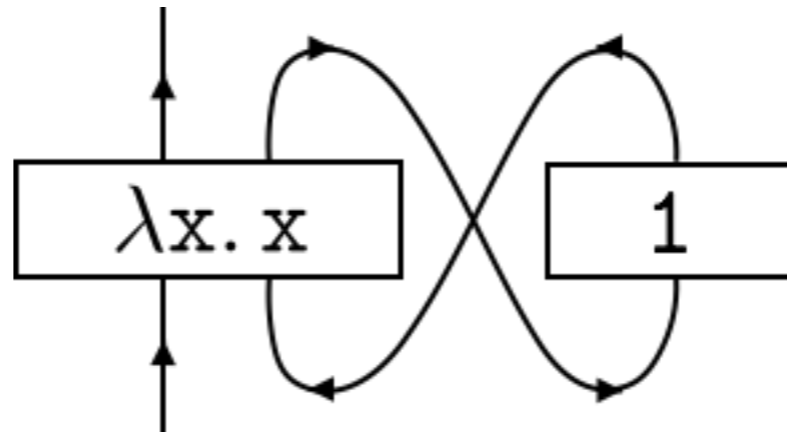
# Token Machine Semantics

**PCF terms**



**token machines**

$(\lambda x. x) 1$



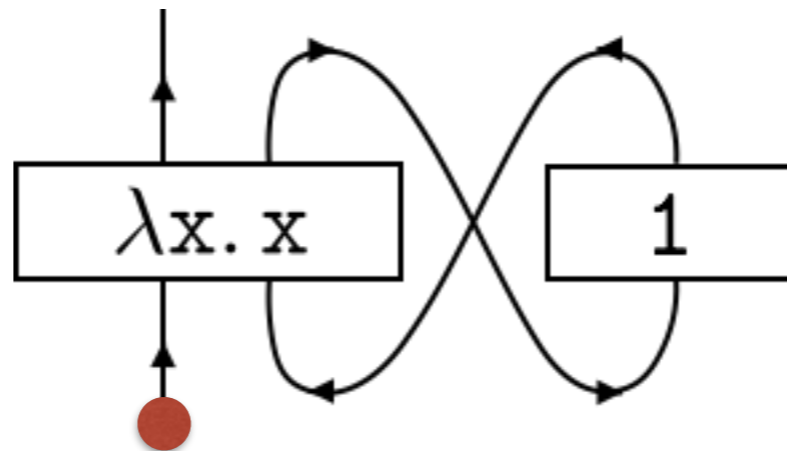
# Token Machine Semantics

**PCF terms**



**token machines**

$(\lambda x. x) 1$



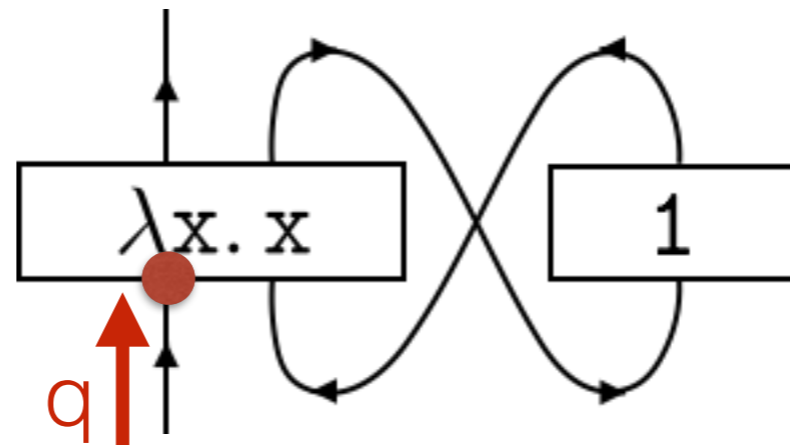
# Token Machine Semantics

**PCF terms**



**token machines**

$(\lambda x. x) 1$



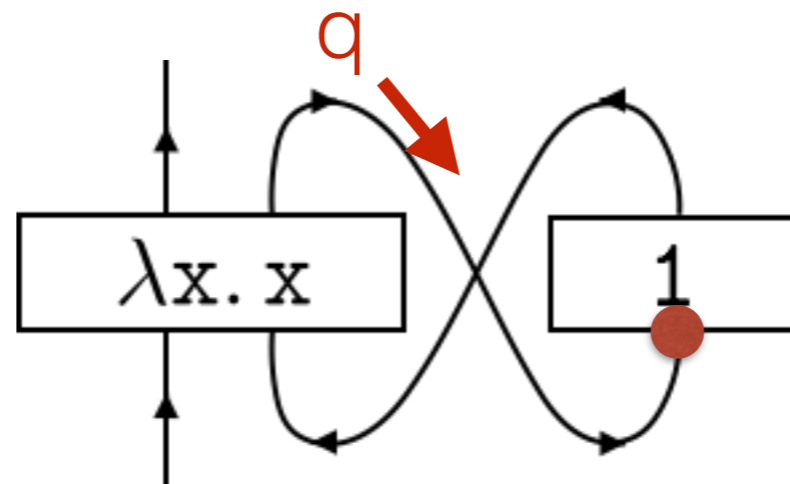
# Token Machine Semantics

**PCF terms**



**token machines**

$(\lambda x. x) 1$



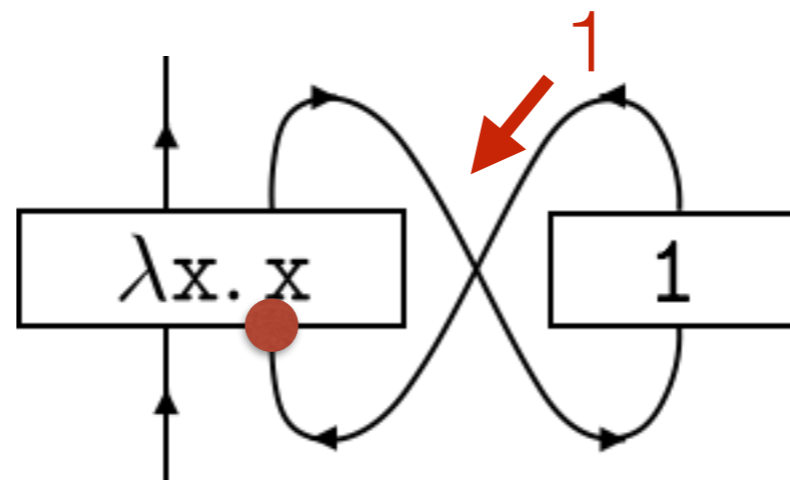
# Token Machine Semantics

**PCF terms**



**token machines**

$(\lambda x. x) 1$



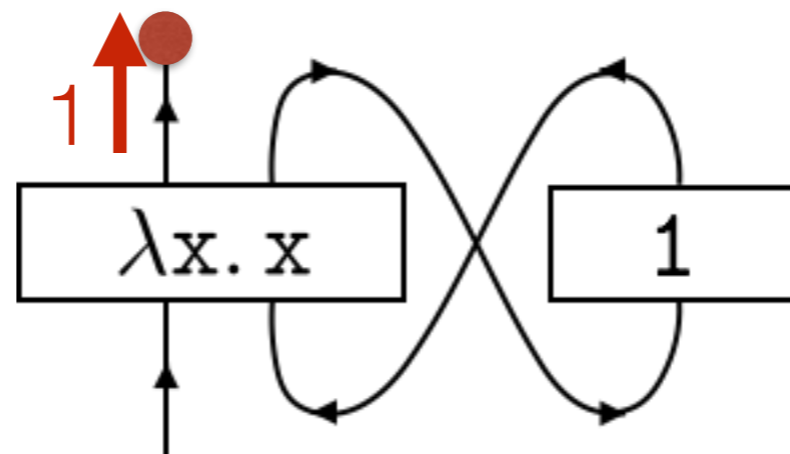
# Token Machine Semantics

**PCF terms**



**token machines**

$(\lambda x. x) 1$





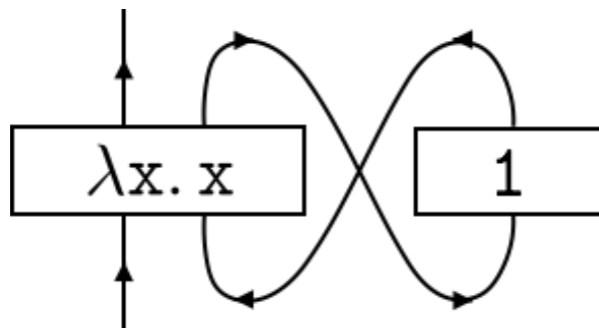
# Token Machine Semantics

**PCF terms**

$(\lambda x. x) 1$



**token  
machines**



- applications to compilation  
[Mackie '95] [Pinto '01] [Ghica '07]
- categorical formalization  
[Abramsky et al. '02]

# Token Machine Semantics

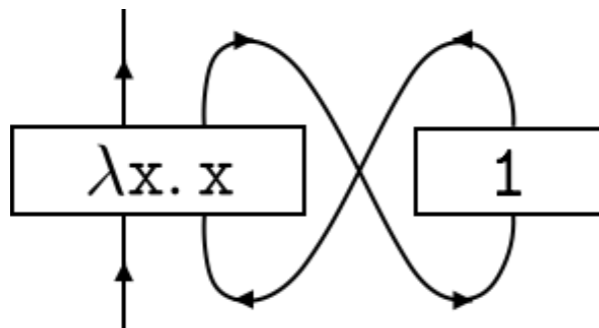
**PCF terms**

**+ computational effects**

$(\lambda x. x) 1$



**token  
machines**



# Token Machine Semantics

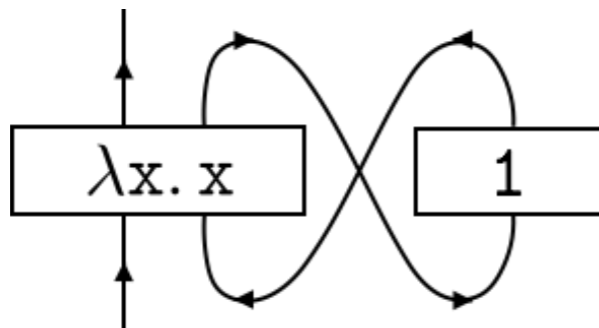
**PCF terms**

recursion

**+ computational effects**

$(\lambda x. x) 1$

**token  
machines**



# Token Machine Semantics

**PCF terms**

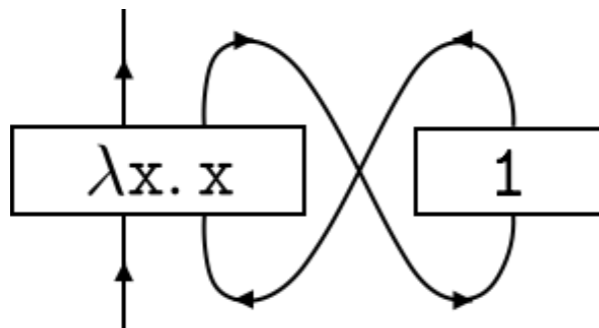
recursion

**+ computational effects**

$(\lambda x. x) 1$

- memoryful GoI [Hoshino, — & Hasuo '14]

**token  
machines**



# Token Machine Semantics

**PCF terms**

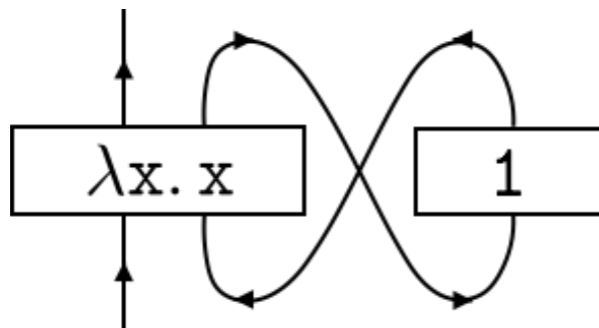
recursion

**+ computational effects**

$(\lambda x. x) 1$

- memoryful Gol [Hoshino, — & Hasuo '14]
- memoryful Gol II

**token  
machines**



# Memoryful GoI

**effectful  
PCF terms**

call-by-value PCF terms

with algebraic operations [Plotkin & Power '01]

- nondeterministic choice
- probabilistic choice
- actions on global states

$\text{choose}(M, N)$

$\text{choose}_p(M, N)$

$\text{lookup}_\ell(M_{v_0}, \dots, M_{v_n})$

$\text{update}_{\ell, v}(M)$

**transducers**

# Memoryful GoI

**effectful  
PCF terms**

call-by-value PCF terms

with algebraic operations [Plotkin & Power '01]



**transducers**

$$(\lambda x. M) V \rightarrow M[V/x]$$

$$\text{rec}(\mathbf{f}, \mathbf{x}). M \rightarrow (\lambda x. M) [\text{rec}(\mathbf{f}, \mathbf{x}). M/\mathbf{f}]$$

$$\text{op}(M_1, \dots, M_n) \xrightarrow{\text{op}_i} M_i$$

$$\text{op}() \downarrow_{\text{op}}$$

# Memoryful GoI

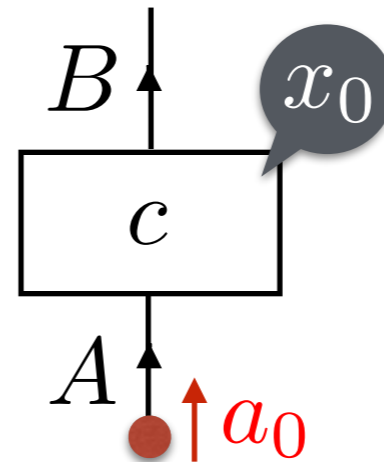
effectful  
PCF terms



transducers

stream transducers (Mealy machines)

$$(X, c: X \times A \rightarrow T(X \times B), x_0 \in X): A \rightarrow B$$





# Memoryful GoI

effectful  
PCF terms



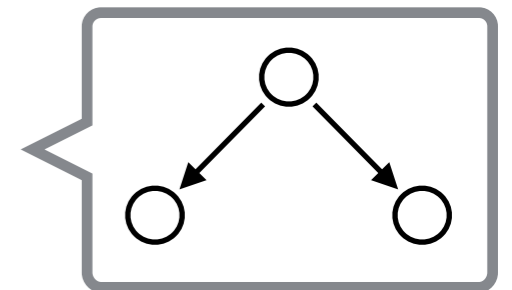
transducers

stream transducers (Mealy machines)

$(X, c: X \times A \rightarrow T(X \times B), x_0 \in X): A \rightarrow B$

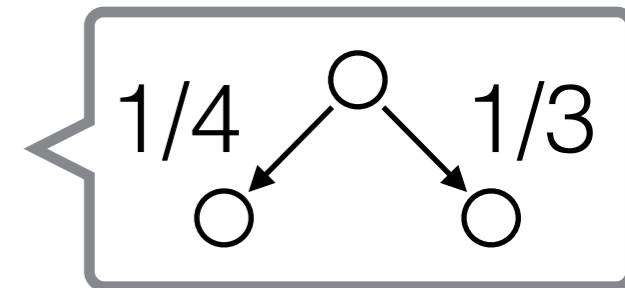
- nondeterministic transitions

$$TX = \mathcal{P}(X)$$



- probabilistic transitions

$$TX = \mathcal{D}_{\leq 1}(X)$$



- transitions with global states

$$TX = (1 + X \times S)^S$$

# Memoryful GoI

**effectful  
PCF terms**



compositional translation

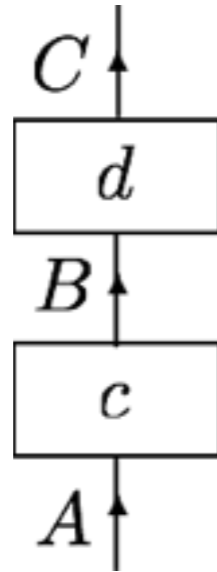
via coalgebraic component calculus

[Barbosa '03] [Hasuo, Jacobs '11]

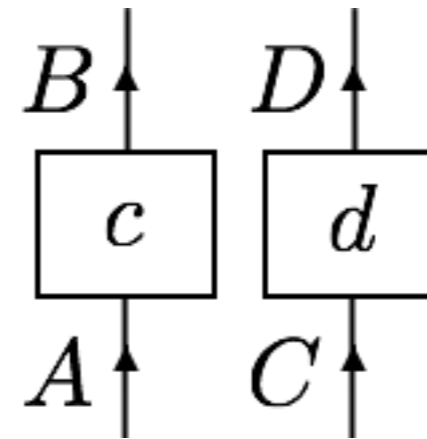
**transducers**

# Component Calculus

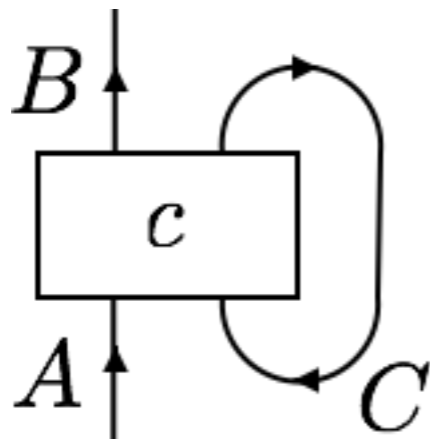
sequential composition



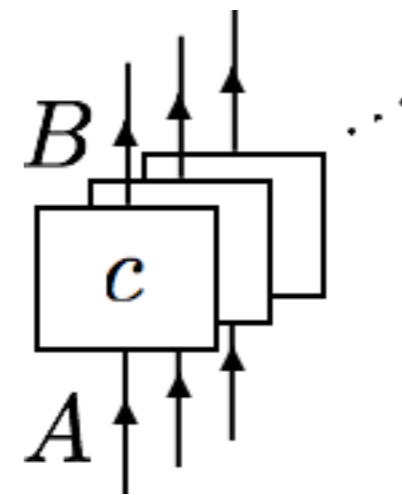
parallel composition



trace operator

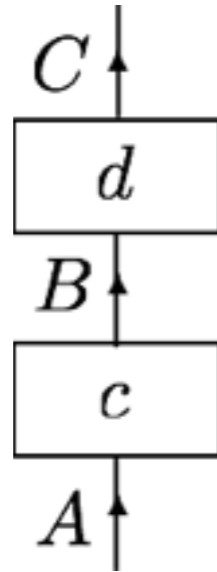


countable copy operator

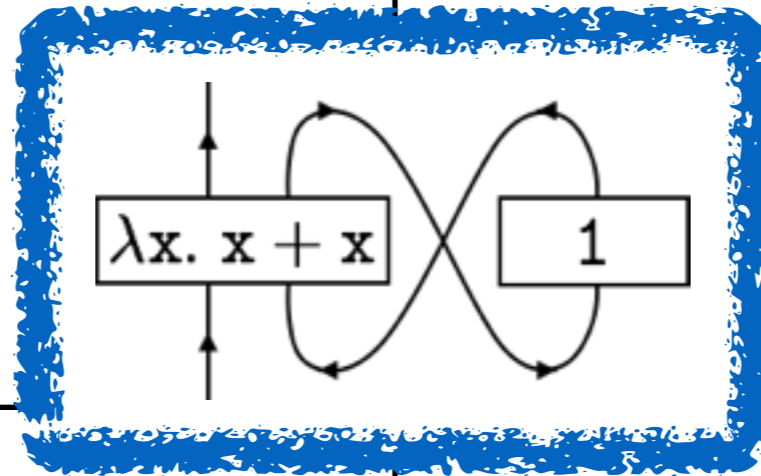
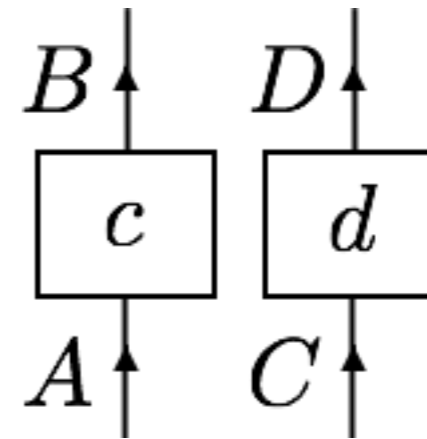


# Component Calculus

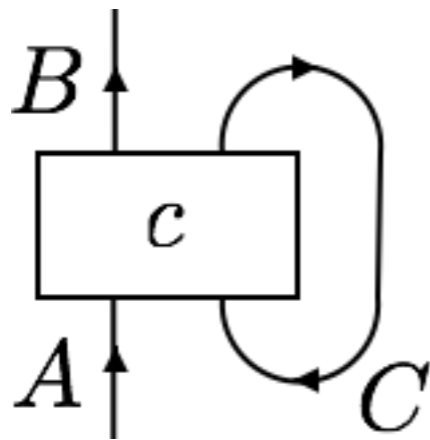
sequential composition



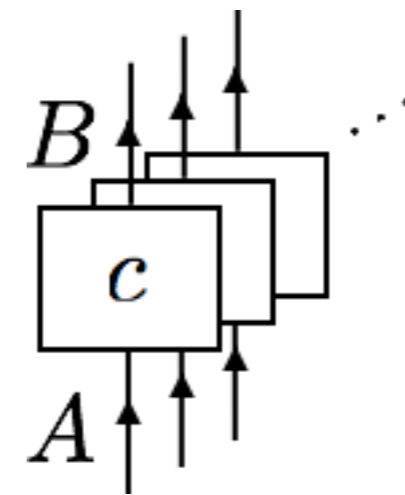
parallel composition



trace operator

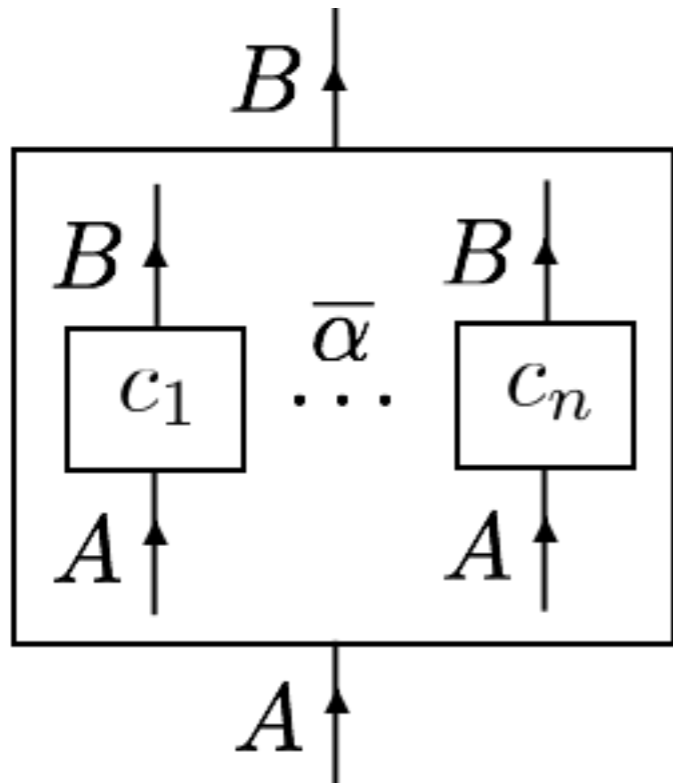


countable copy operator

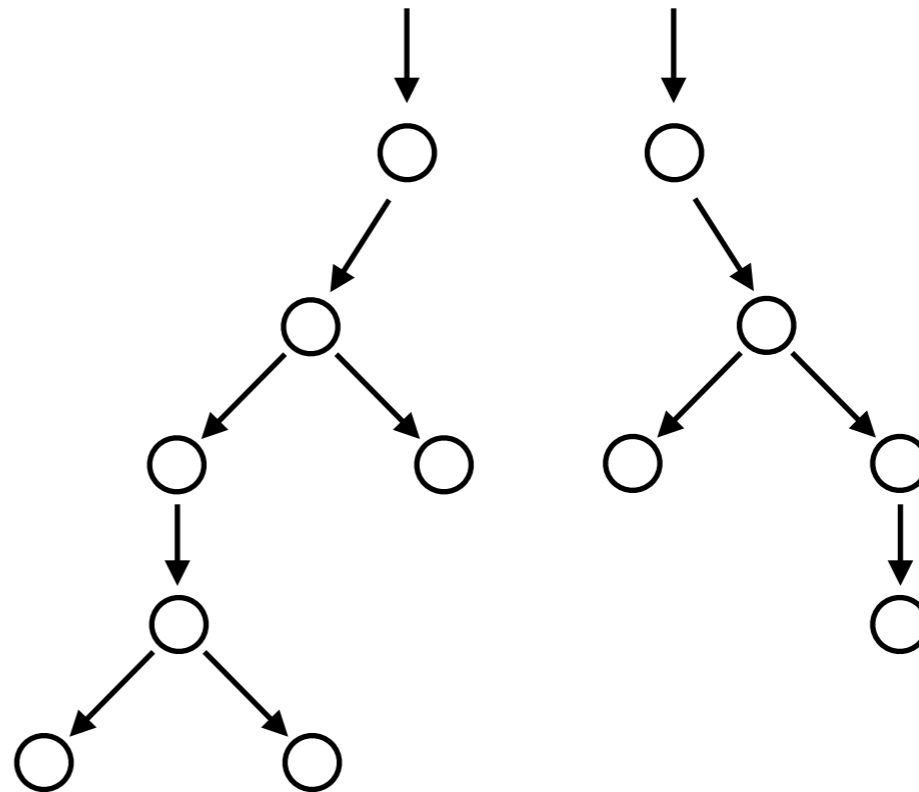


# Component Calculus for Effects

lifted algebraic operations

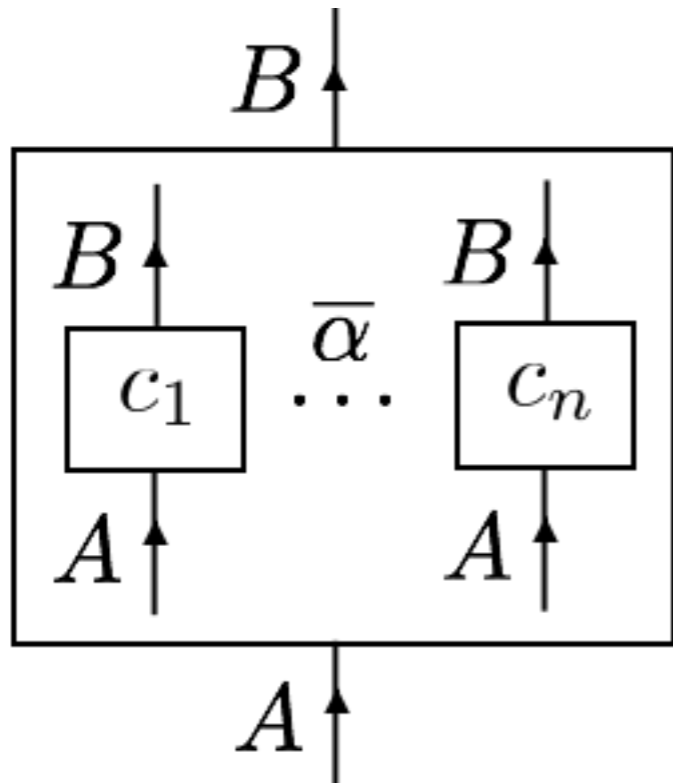


$\alpha = \text{choose}_{0.4}$

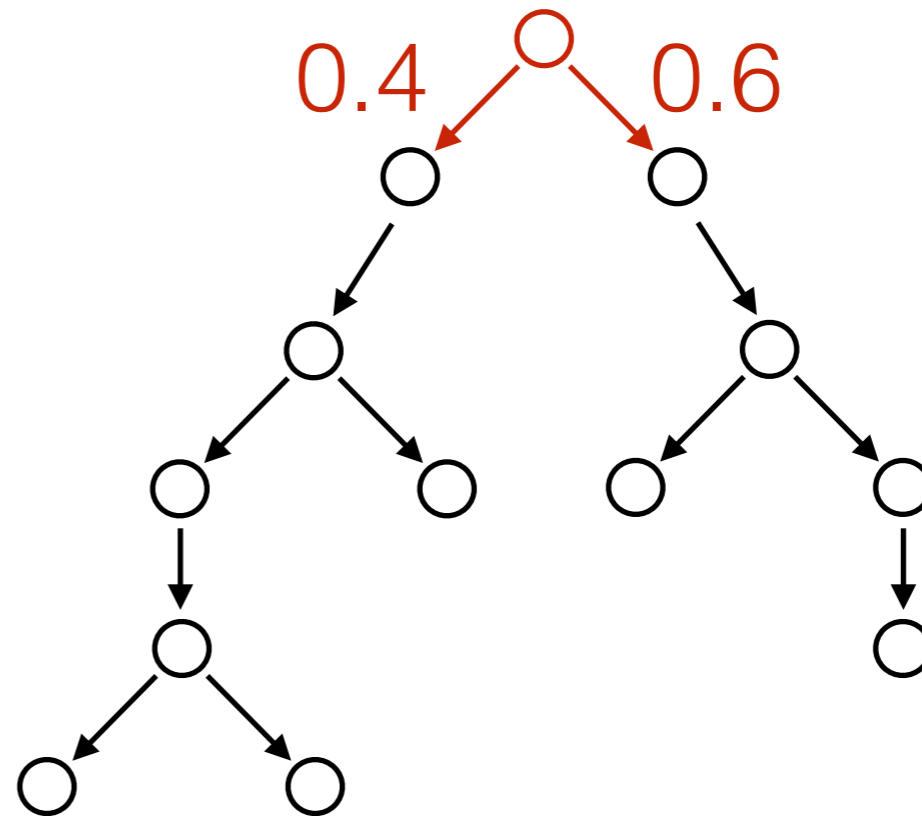


# Component Calculus for Effects

lifted algebraic operations

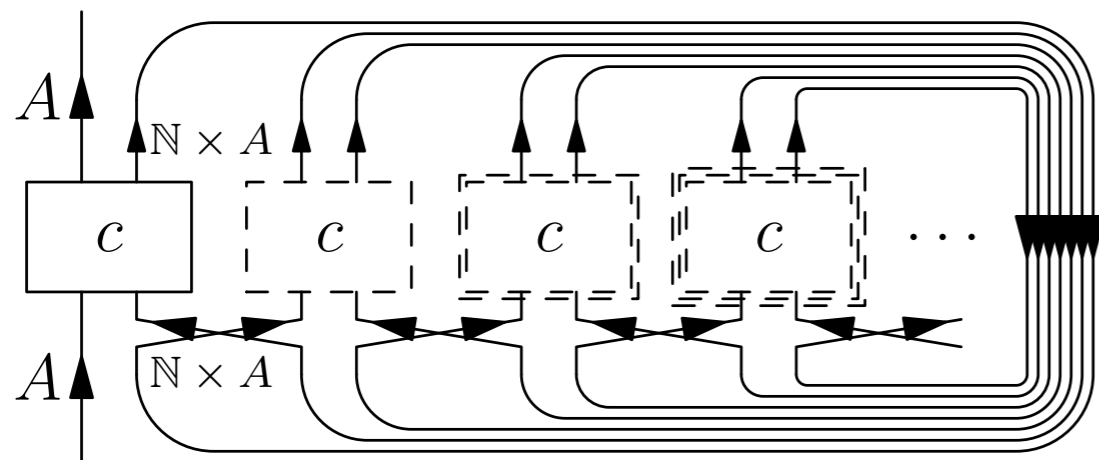


$\alpha = \text{choose}_{0.4}$

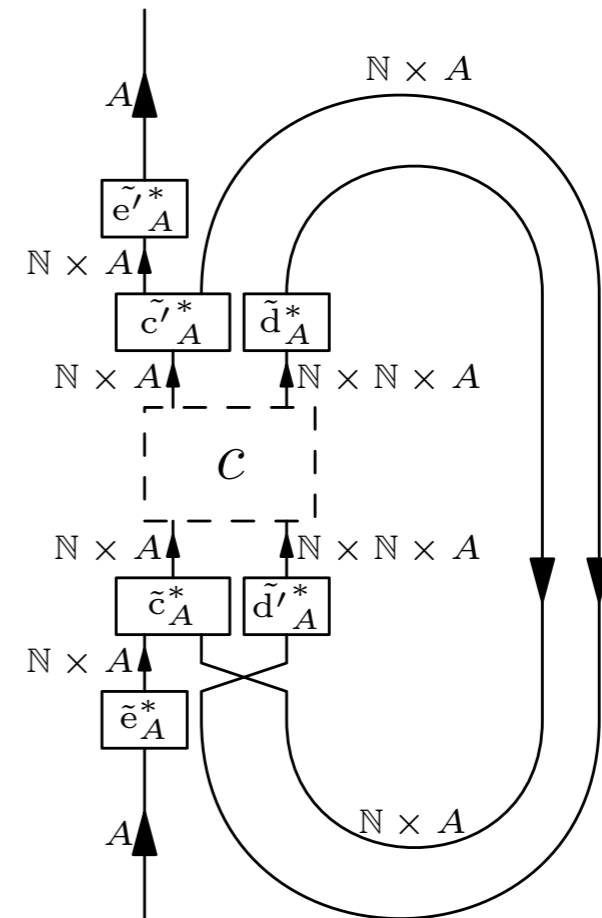


# Component Calculus for Recursion

Girard style  
fixed point operator

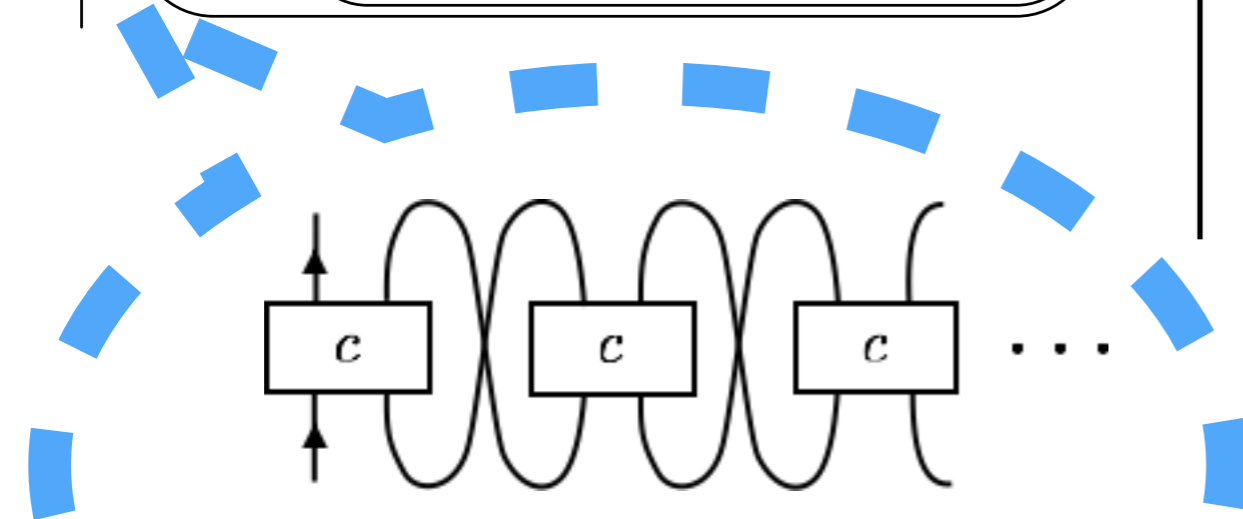
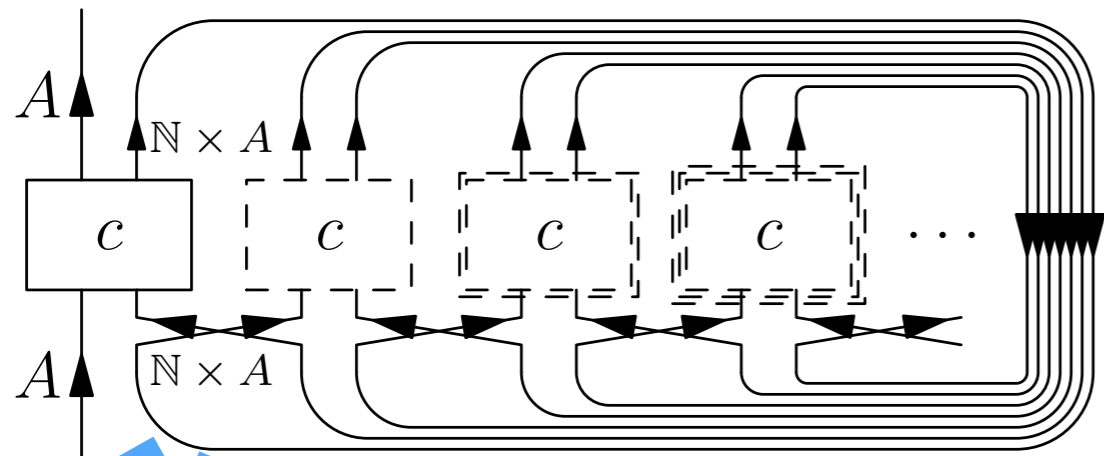


Mackie style  
fixed point operator



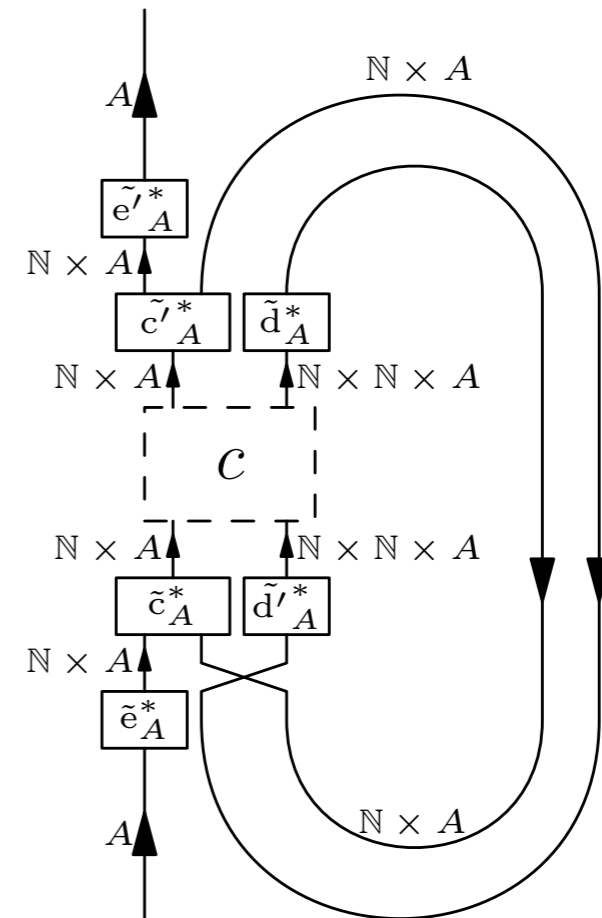
# Component Calculus for Recursion

Girard style  
fixed point operator



$$\text{fix}(F) = F(F(F(\dots)))$$

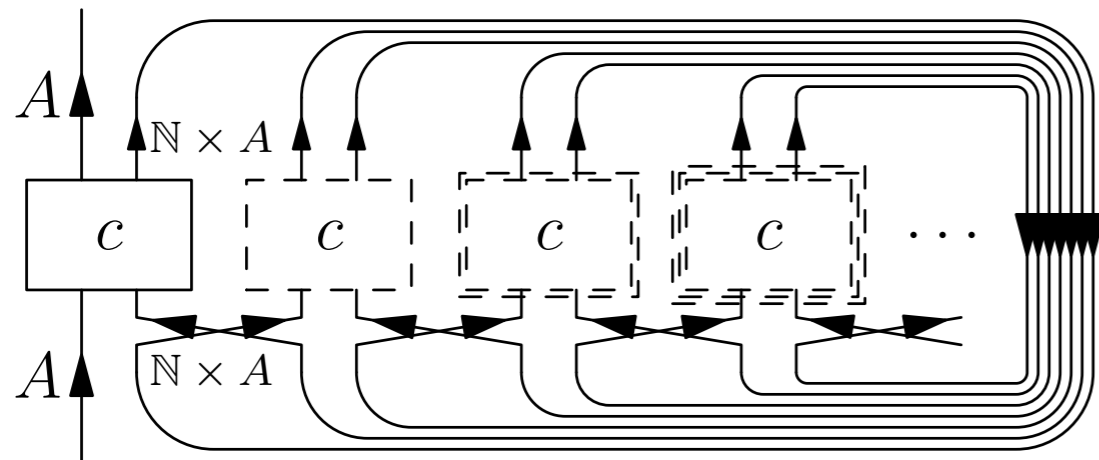
Mackie style  
fixed point operator



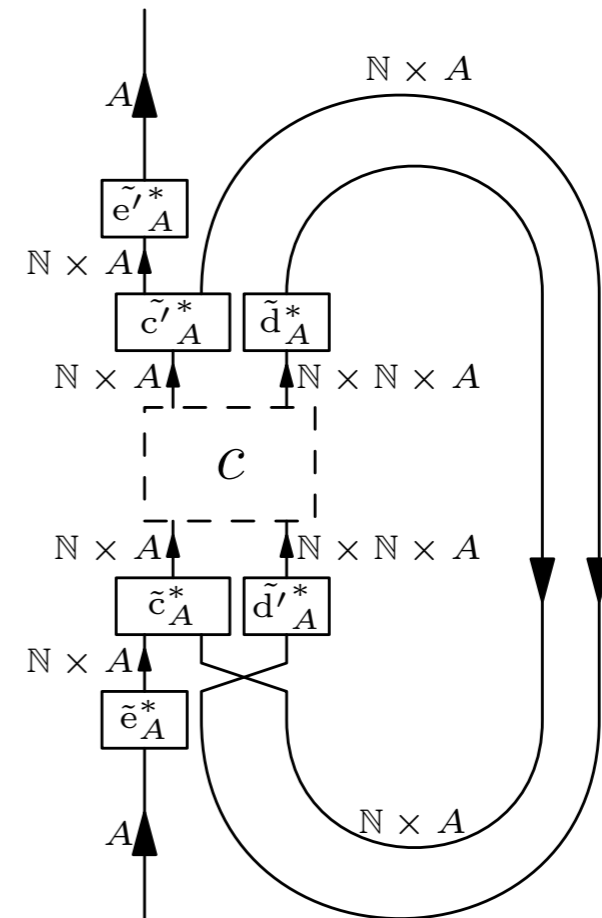


# Component Calculus for Recursion

Girard style  
fixed point operator

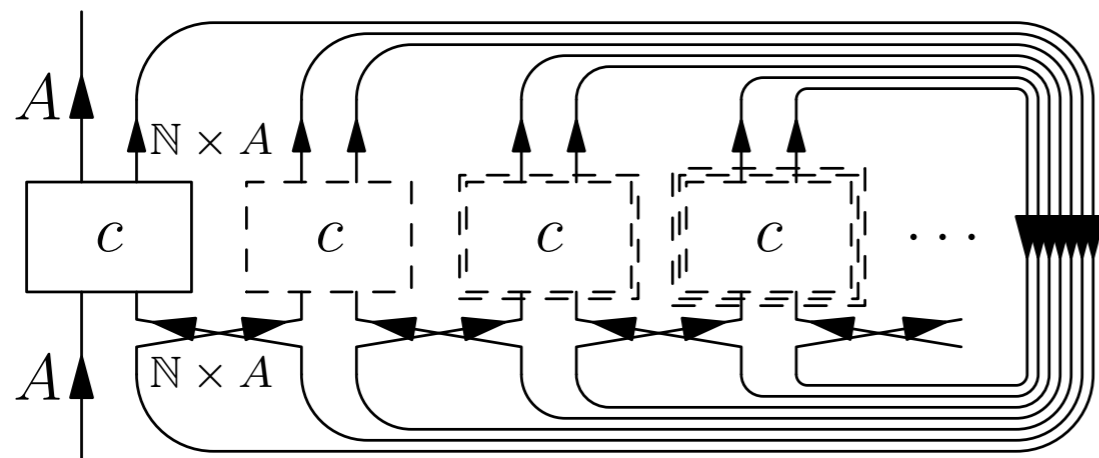


Mackie style  
fixed point operator

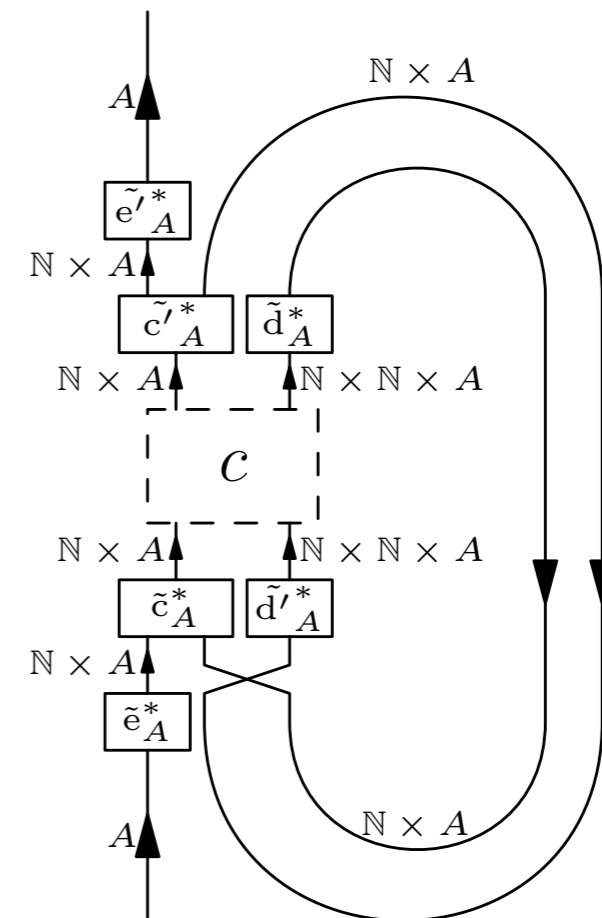


# Component Calculus for Recursion

Girard style  
fixed point operator



Mackie style  
fixed point operator



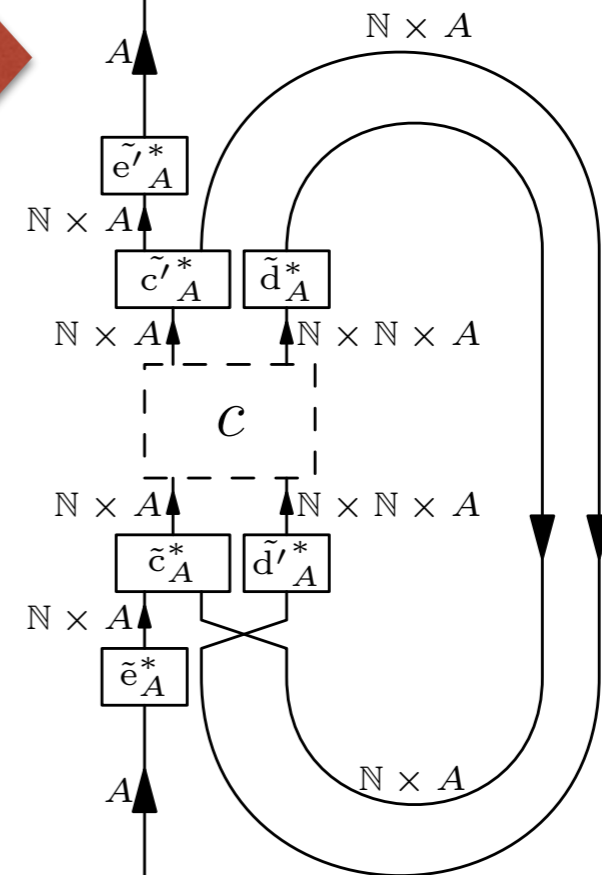
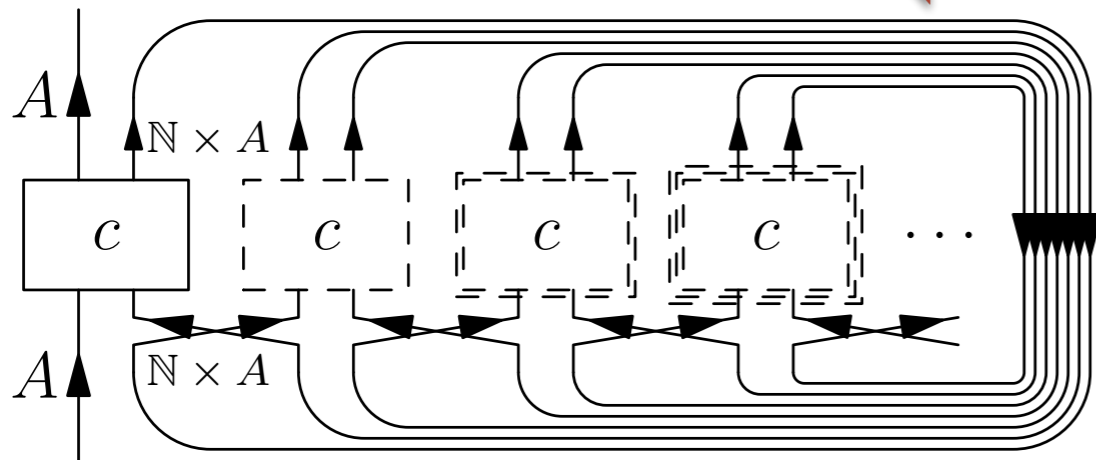
- fixed point wrt. “cross connection”
- supremum of finite approximations

# Component Calculus for Recursion

Girard style  
fixed point operator

Mackie style  
fixed point operator

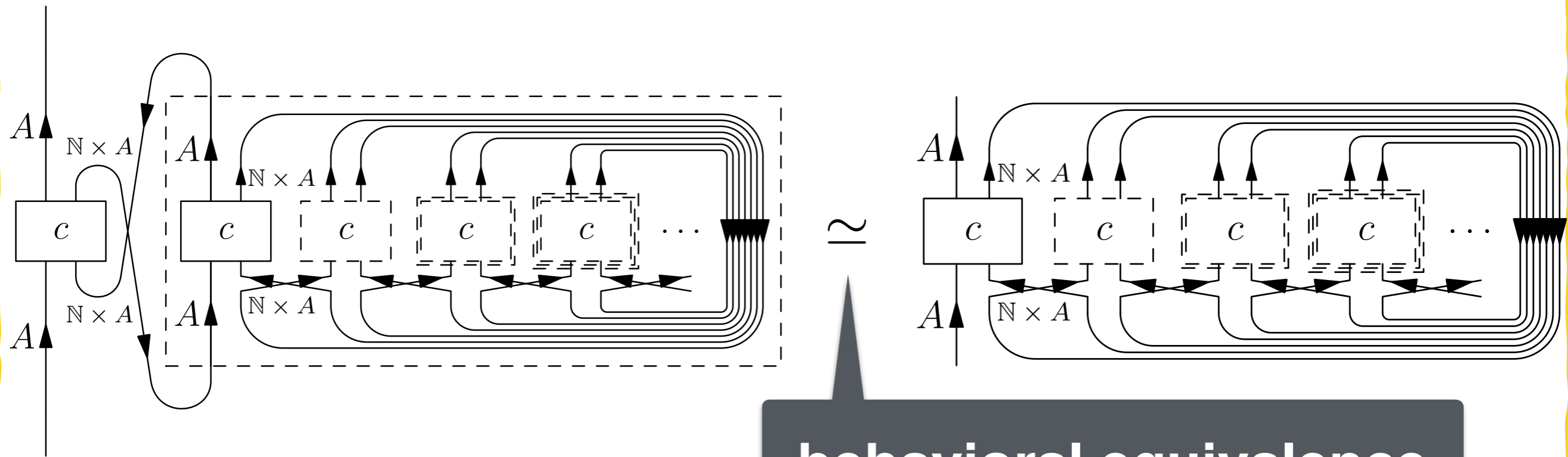
coincidence



- fixed point wrt. “cross connection”
- supremum of finite approximations

# Component Calculus for Recursion

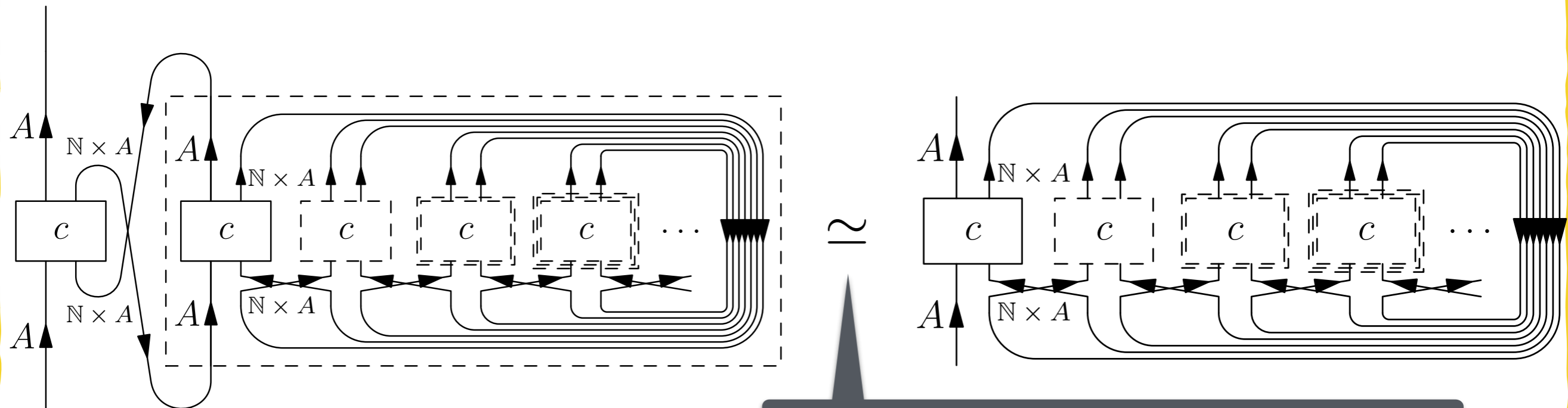
**Prop.** (Girard style as **fixed point** wrt. “cross connection”)



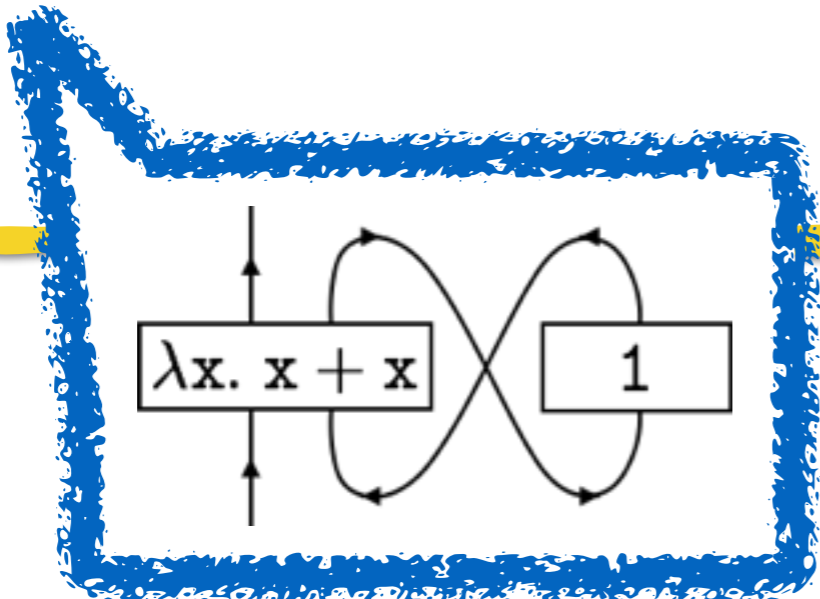
behavioral equivalence

# Component Calculus for Recursion

**Prop.** (Girard style as **fixed point** wrt. “cross connection”)

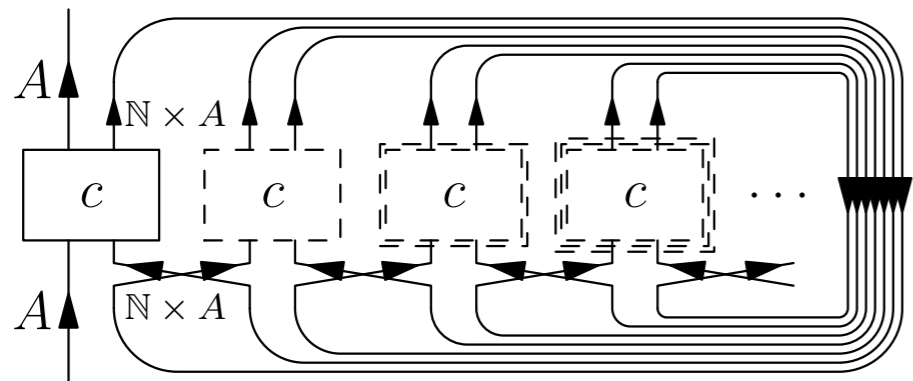


behavioral equivalence

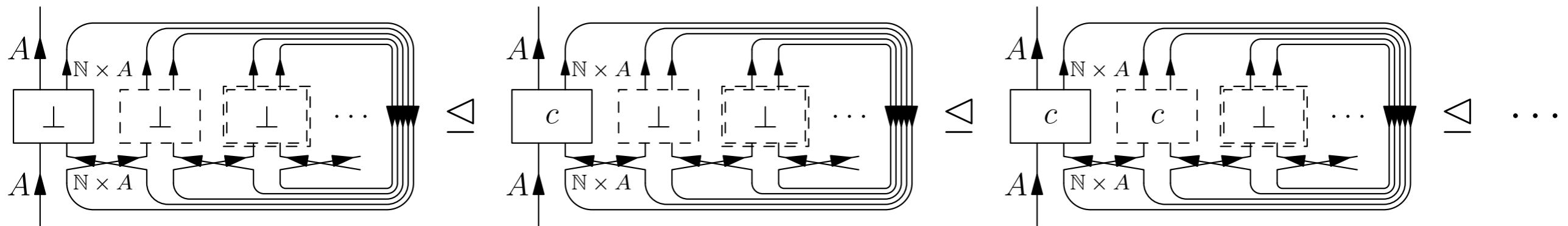


# Component Calculus for Recursion

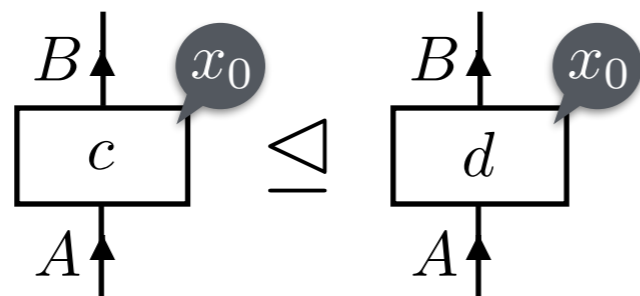
**Prop.** (Girard style as **supremum** of finite approximations)



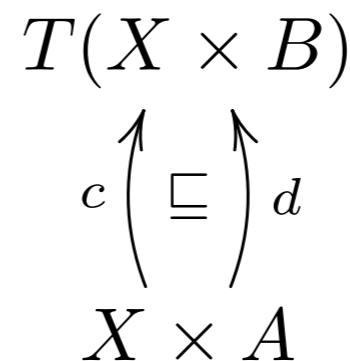
is the supremum of the  $\omega$ -chain



where

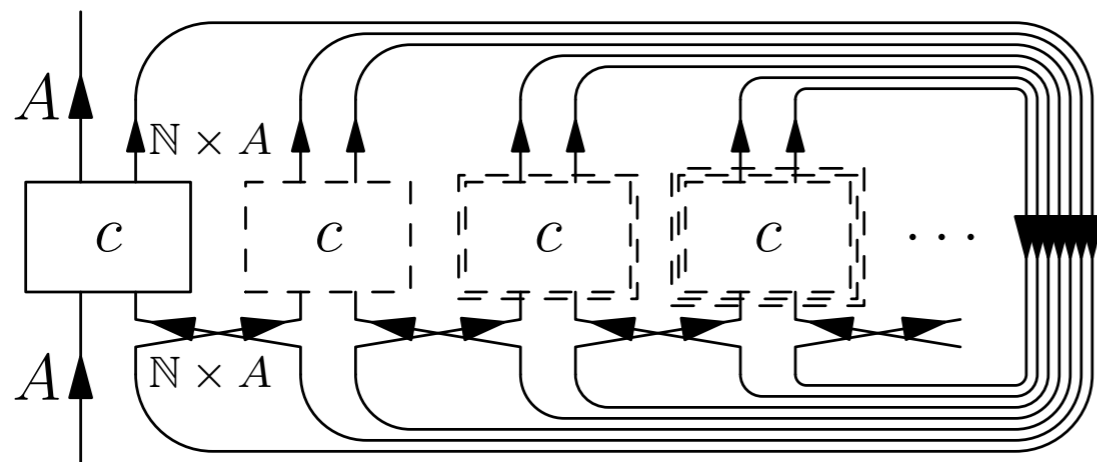


def.

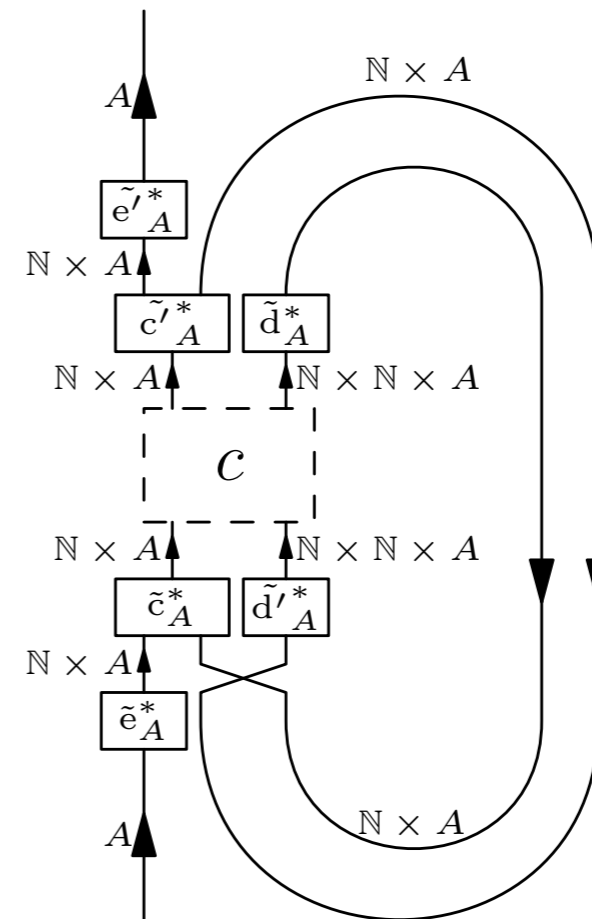


# Component Calculus for Recursion

**Thm. (coincidence of Girard style & Mackie style)**



$\approx$



# Memoryful GoI

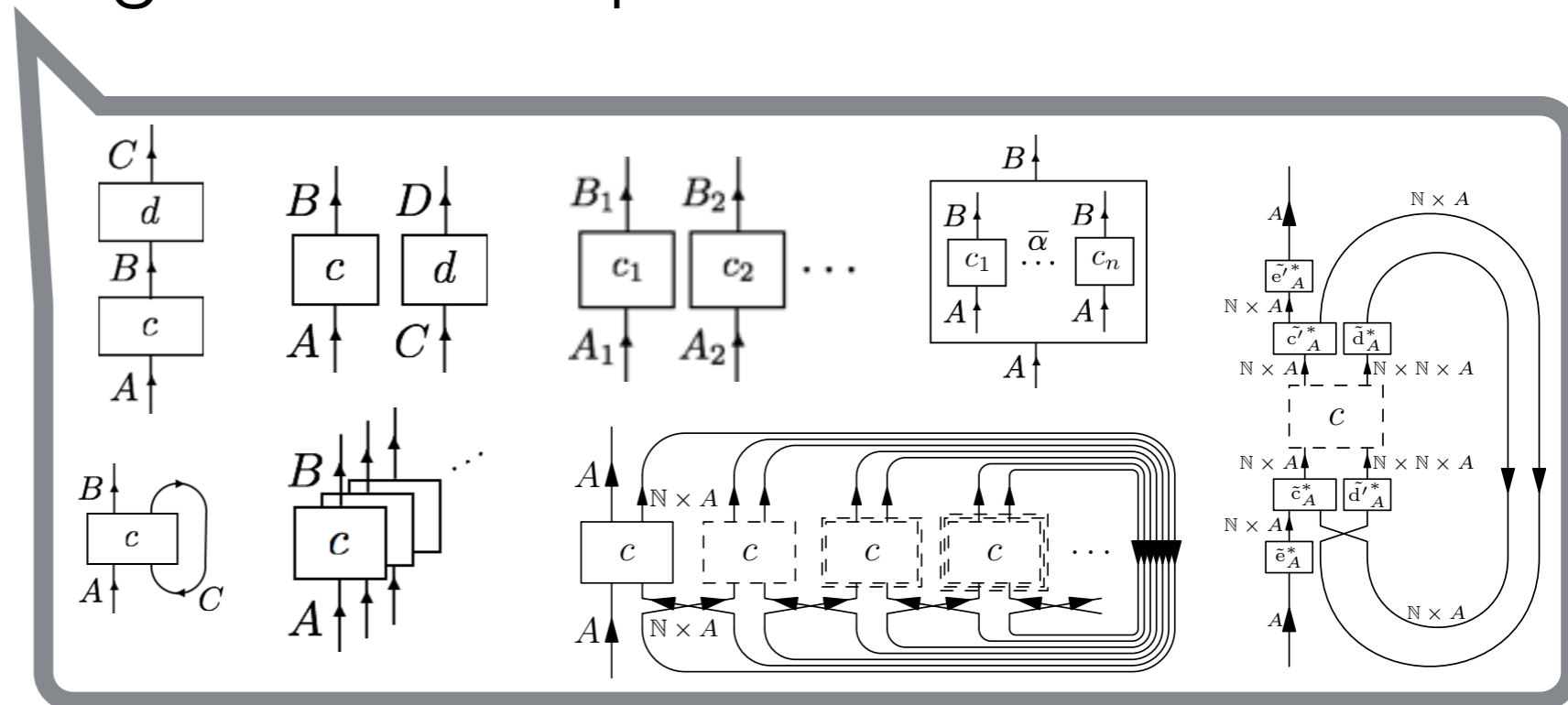
effectful  
PCF terms



transducers

compositional translation

via coalgebraic component calculus





# Translation

**Def. (translation  $(\Gamma \vdash M : \tau)$ )**

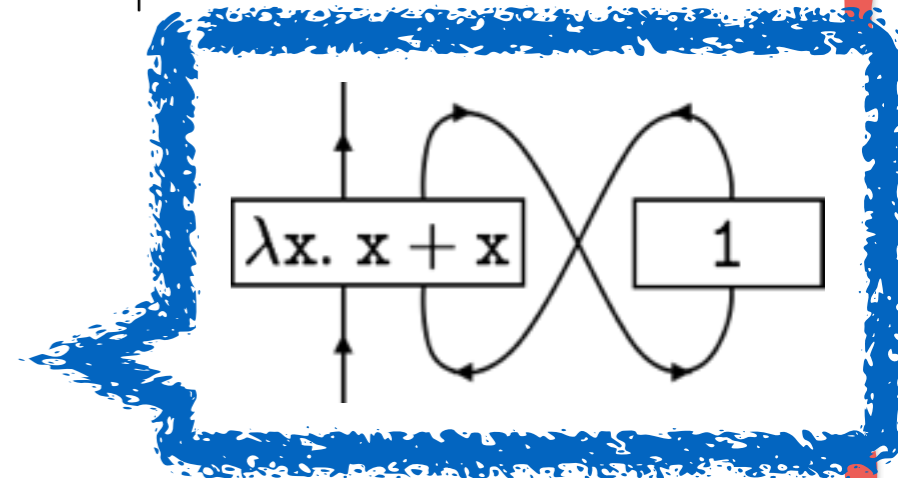
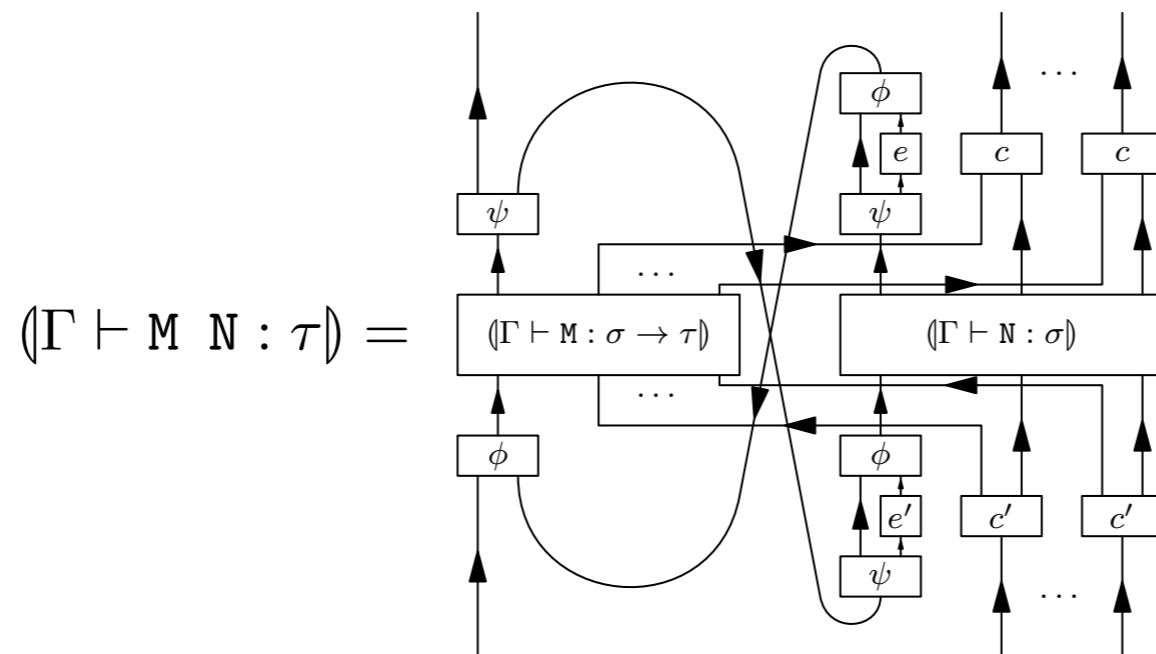
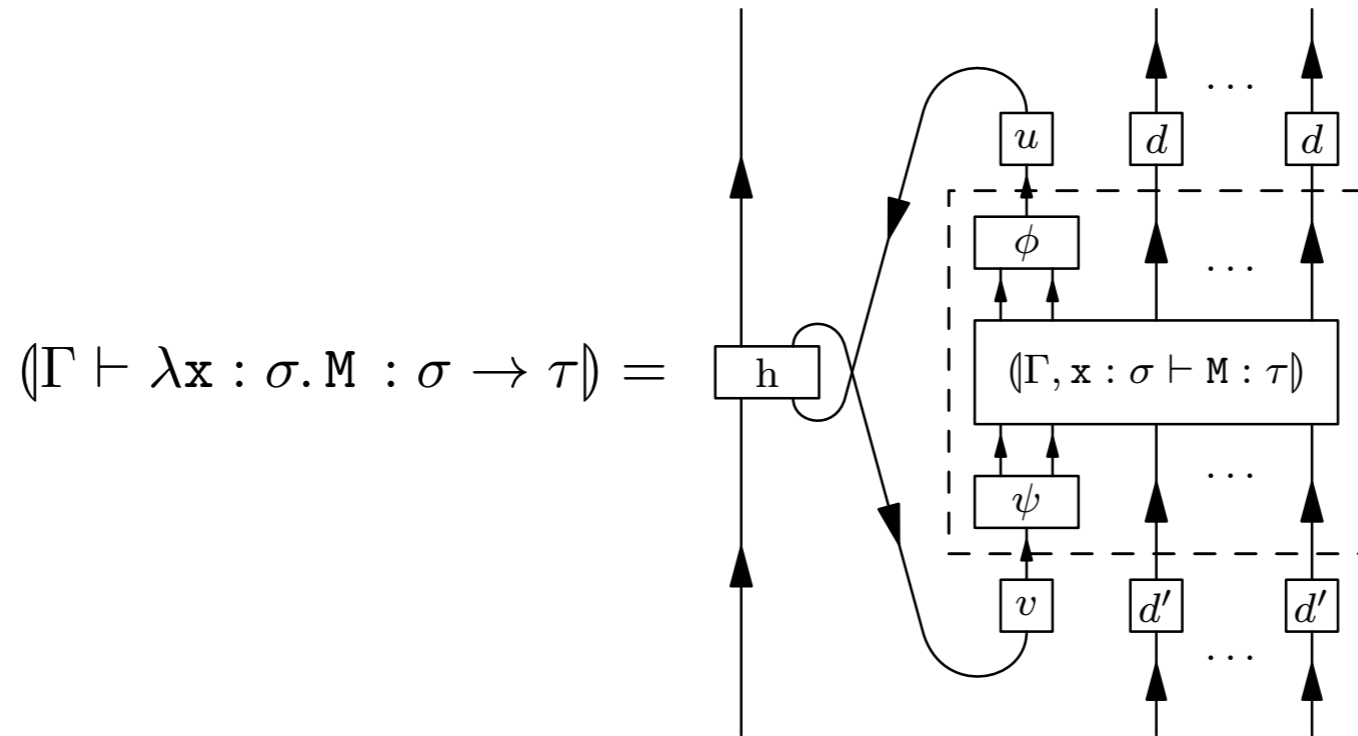
For a type judgement  $(\Gamma \vdash M : \tau)$  ( $\Gamma = \mathbf{x}_1 : \tau_1, \dots, \mathbf{x}_n : \tau_n$ )

we inductively define

$$(\Gamma \vdash M : \tau) = \begin{array}{c} \overbrace{\phantom{N \uparrow N \uparrow \dots \uparrow N}}^n \\ N \uparrow N \uparrow \dots \uparrow N \\ \boxed{(\Gamma \vdash M : \tau)} \\ N \uparrow N \uparrow \dots \uparrow N \end{array}$$

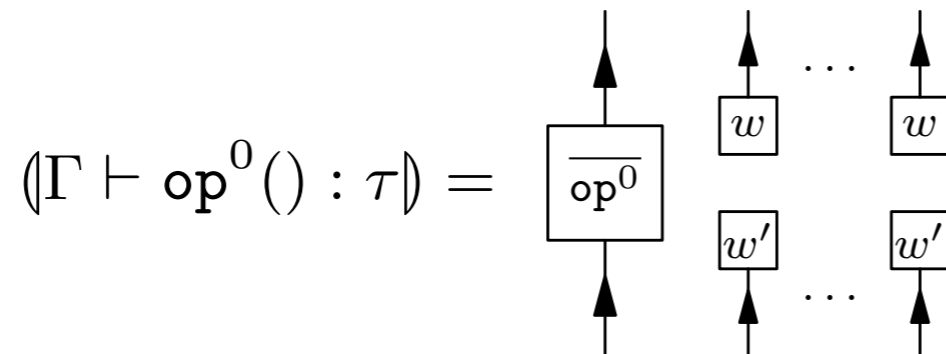
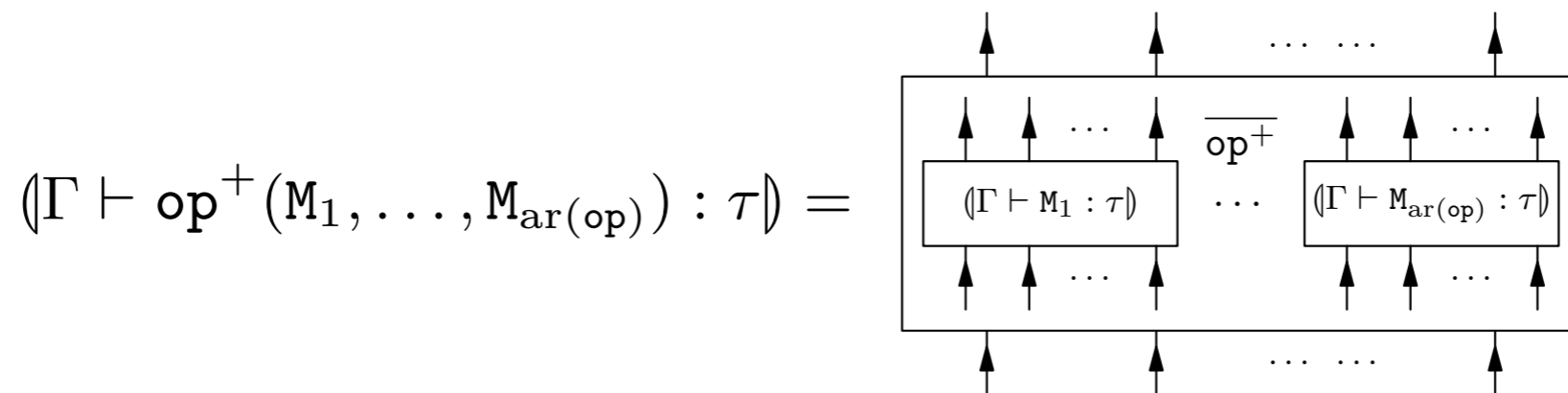
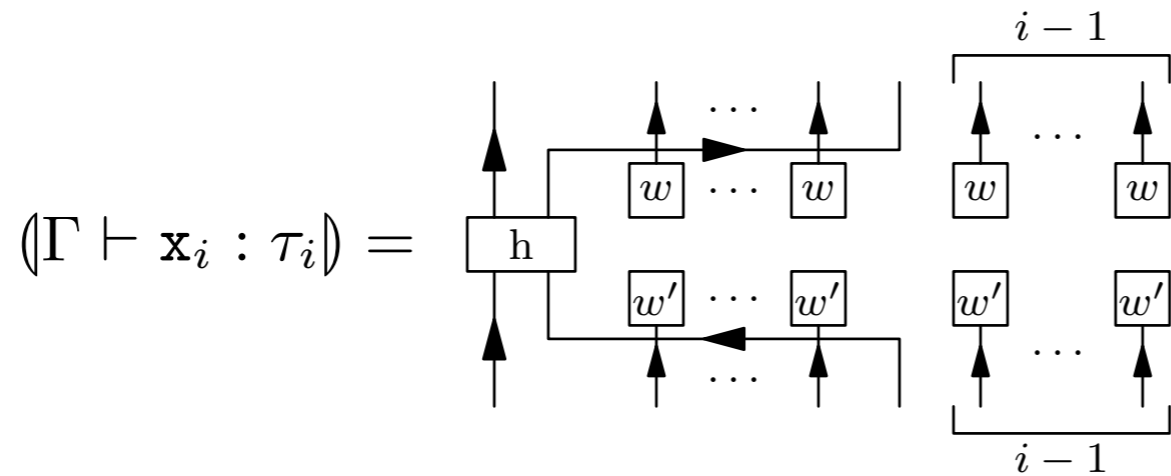
# Translation

## Def. (translation $(\Gamma \vdash M : \tau)$ )



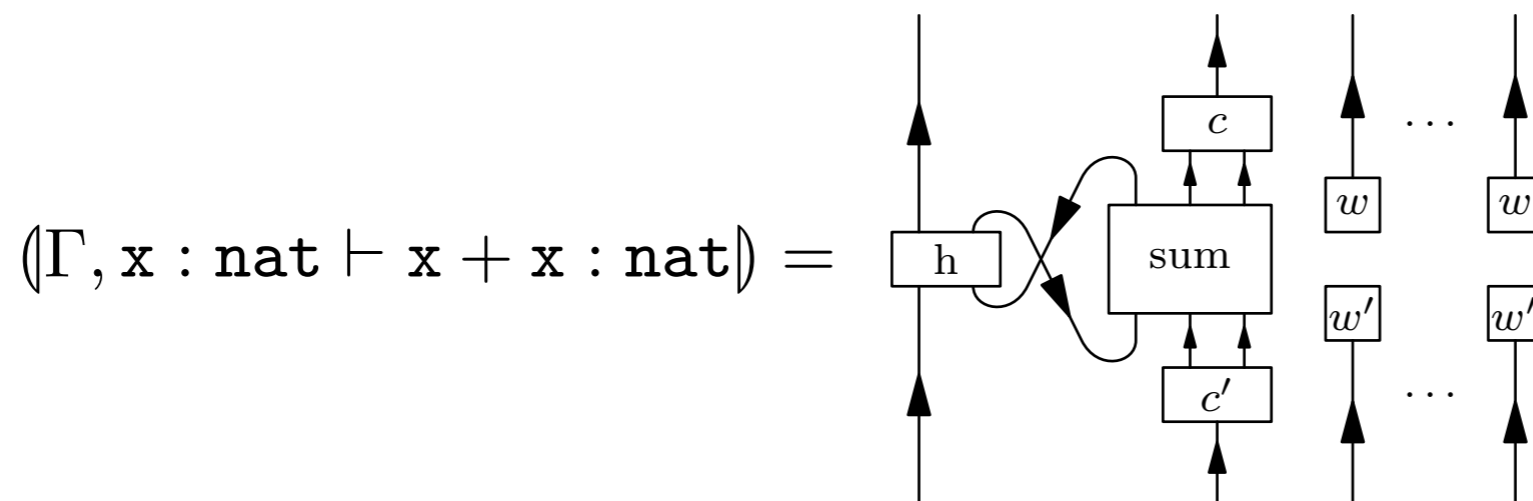
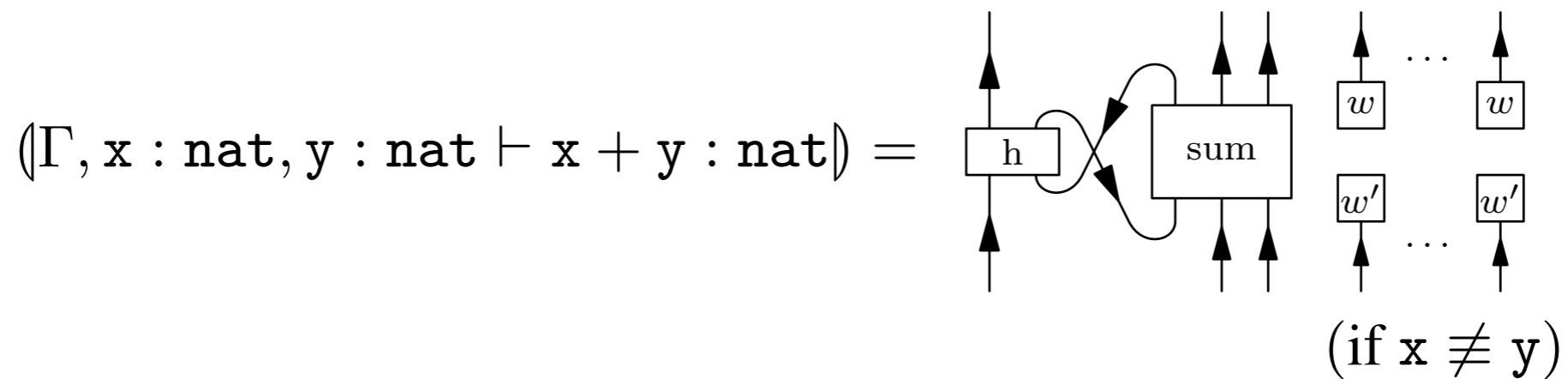
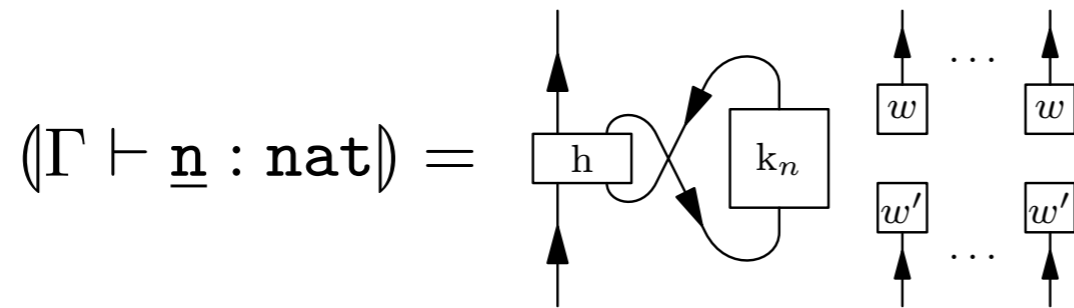
# Translation

## Def. (translation $(\Gamma \vdash M : \tau)$ )



# Translation

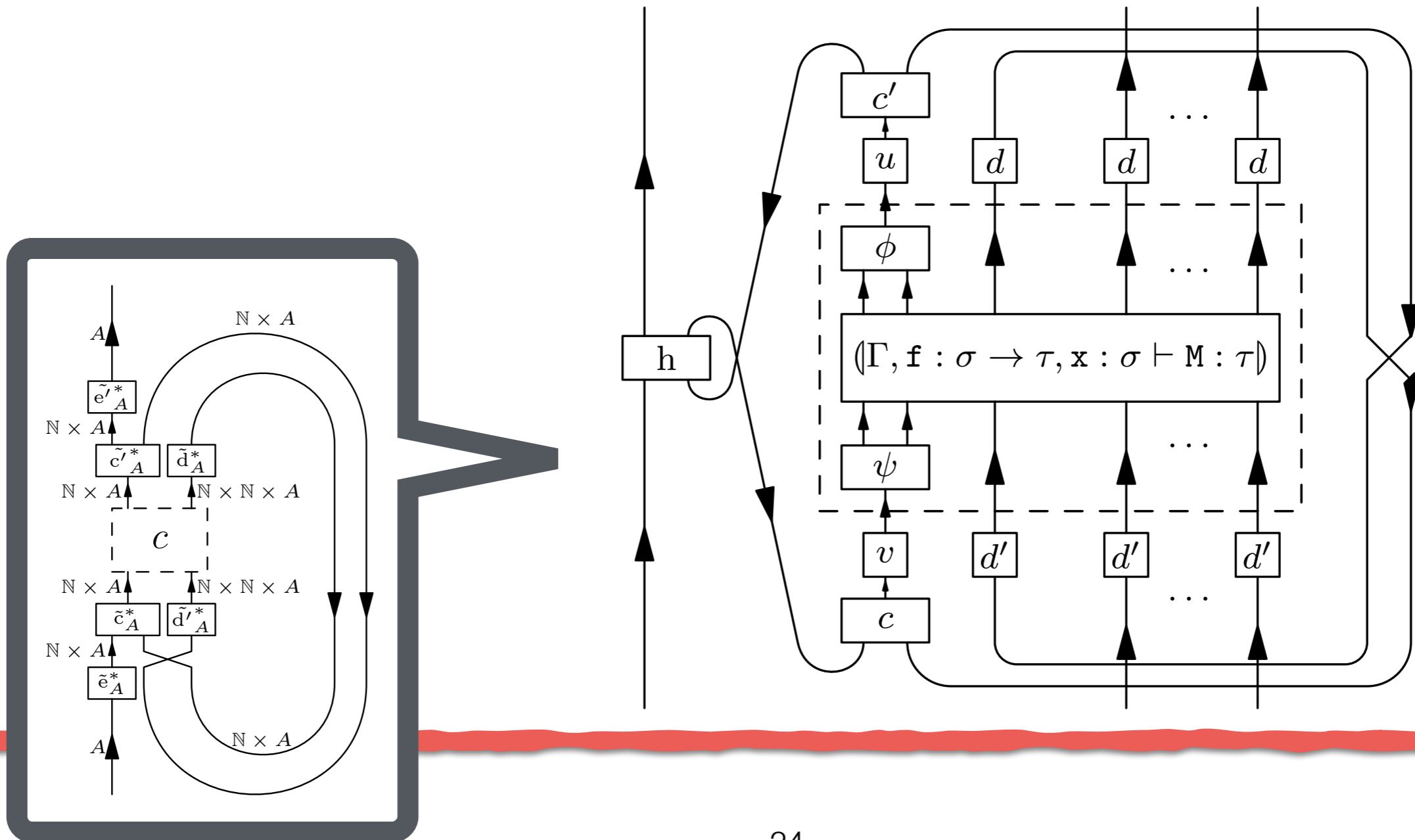
## Def. (translation $(\Gamma \vdash M : \tau)$ )



# Translation

**Def. (translation  $(\Gamma \vdash M : \tau)$ )**

$$(\Gamma \vdash \text{rec}(f : \sigma \rightarrow \tau, x : \sigma). M : \sigma \rightarrow \tau) =$$

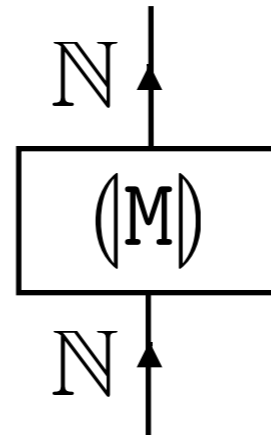


# Translation

**Theorem 6.4** (adequacy). Any closed term  $M$  of base type  $\text{nat}$  satisfies  $\llbracket M \rrbracket = (M)^\dagger$ .

evaluation result  
of term  $M$

output of transducer

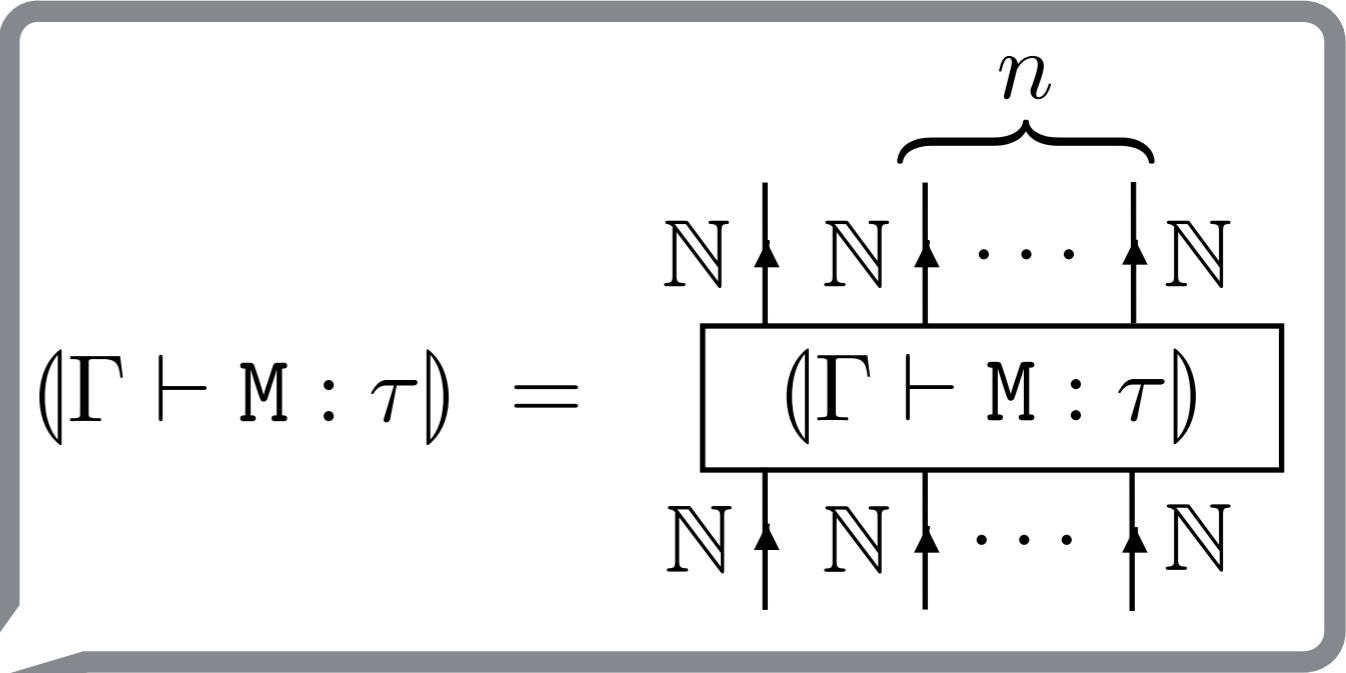


# Memoryful GoI

**effectful  
PCF terms**

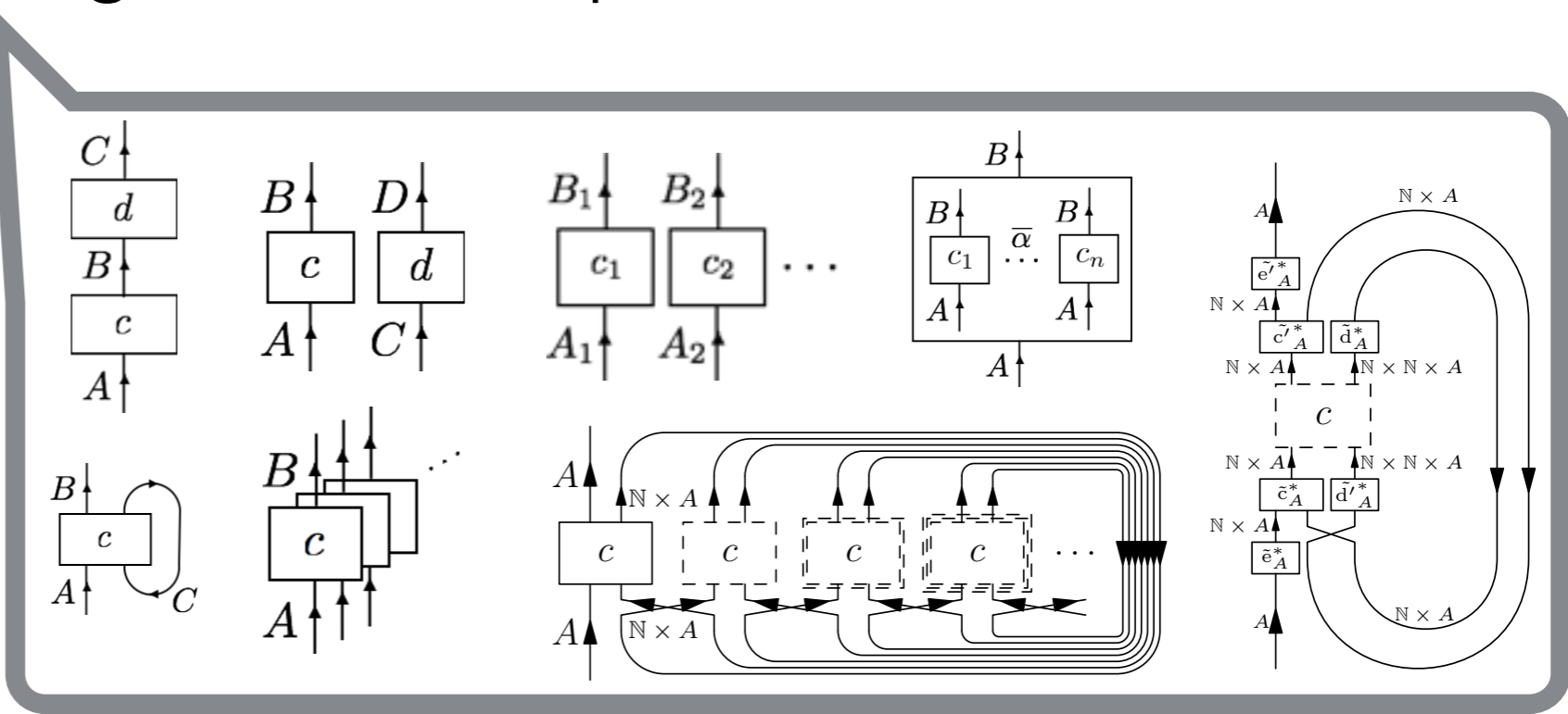


**transducers**



adequate translation

via coalgebraic component calculus



# Example: Recursive Probabilistic Program

```
(rec(flipLoop, x). choose0.4(x, flipLoop(x + 1))) 0
```

coin flips	return value	probability
H	0	0.4
TH	1	$0.4 * 0.6^2$
TTH	2	$0.4 * 0.6^3$
TTTH	3	$0.4 * 0.6^4$



# Example: Recursive Probabilistic Program

```
(rec(flipLoop, x). choose0.4(x, flipLoop(x + 1))) 0
```

coin flips		probability
H	0	0.4
TH	1	$0.4 * 0.6^2$
TTH	2	$0.4 * 0.6^3$
TTTH	3	$0.4 * 0.6^4$

# Example: Recursive Probabilistic Program

```
(rec(flipLoop, x). choose0.4(x, flipLoop(x + 1))) 0
```

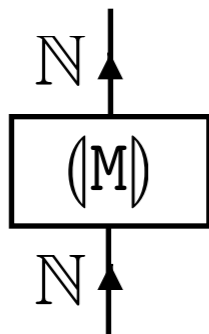
$$\begin{array}{l} \text{evaluation result} \\ \text{of term } M \end{array} = \left[ \begin{array}{l} 0 \mapsto 0.4 \\ 1 \mapsto 0.4 \times 0.6 \\ 2 \mapsto 0.4 \times 0.6^2 \\ 3 \mapsto 0.4 \times 0.6^3 \\ \vdots \end{array} \right] \in \mathcal{D}_{\leq 1}(\mathbb{N})$$

# Example: Recursive Probabilistic Program

```
(rec(flipLoop, x). choose0.4(x, flipLoop(x + 1))) 0
```

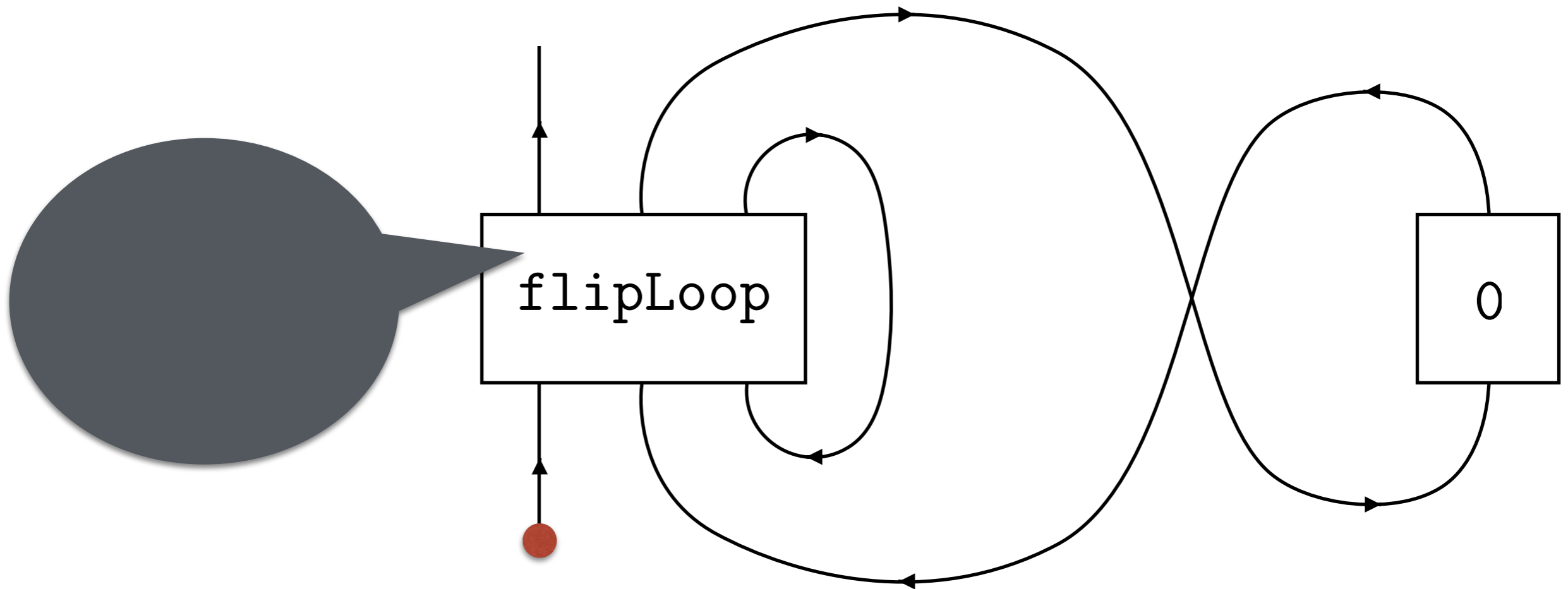
$$\begin{array}{l} \text{evaluation result} \\ \text{of term } M \\ \parallel \end{array} = \left[ \begin{array}{l} 0 \mapsto 0.4 \\ 1 \mapsto 0.4 \times 0.6 \\ 2 \mapsto 0.4 \times 0.6^2 \\ 3 \mapsto 0.4 \times 0.6^3 \\ \vdots \end{array} \right] \in \mathcal{D}_{\leq 1}(\mathbb{N})$$

output of transducer



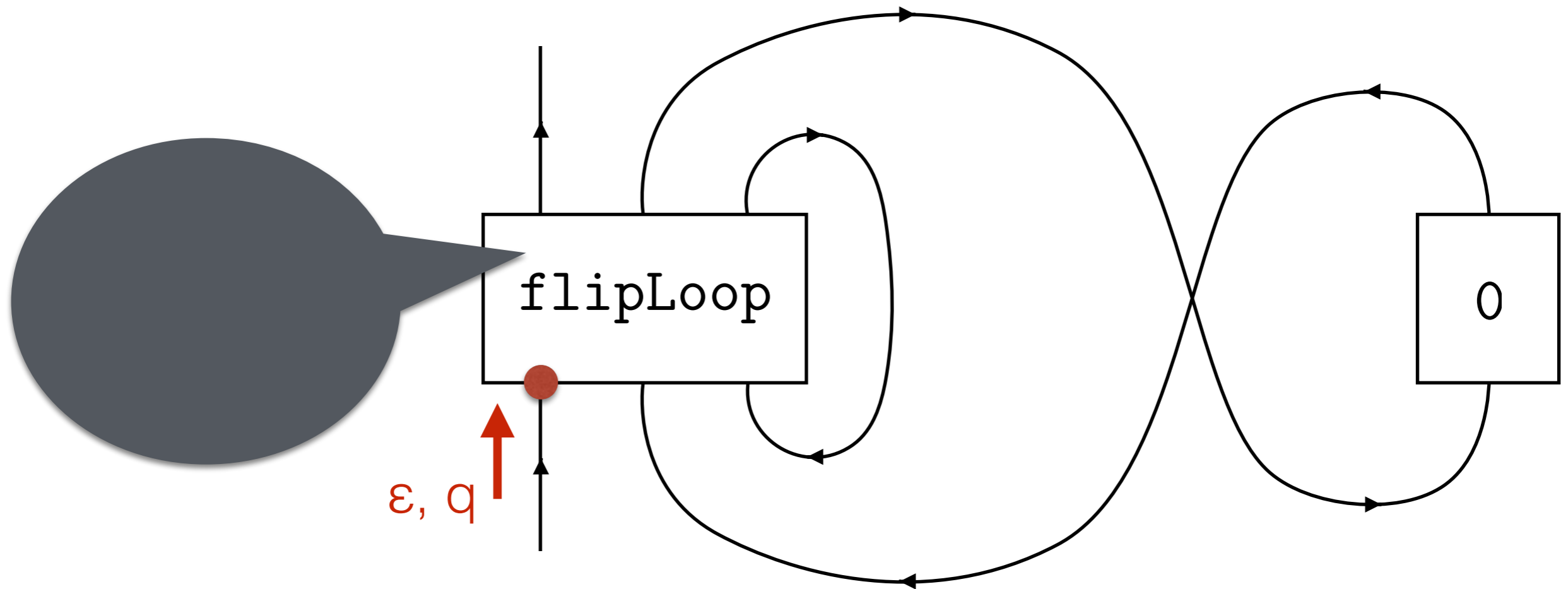
# Example: Recursive Probabilistic Program

```
(rec(flipLoop, x). choose0.4(x, flipLoop(x + 1))) 0
```



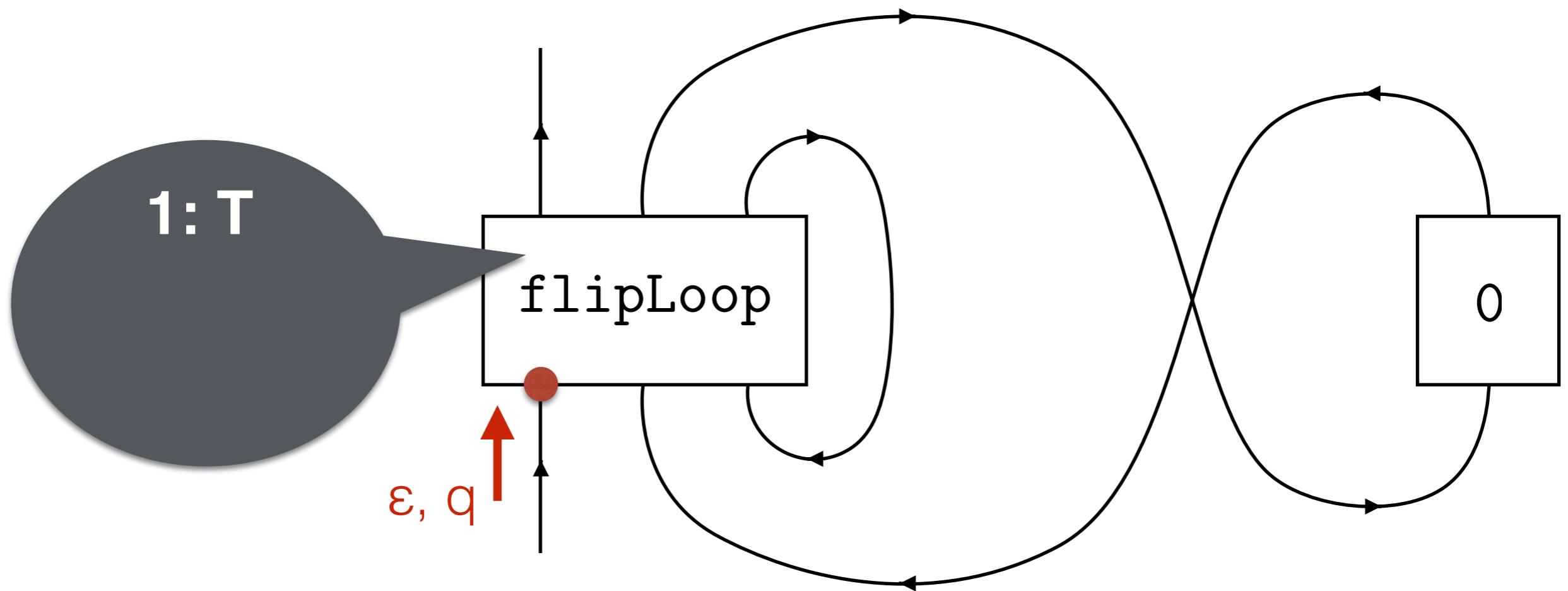
# Example: Recursive Probabilistic Program

```
(rec(flipLoop, x). choose0.4(x, flipLoop(x + 1))) 0
```



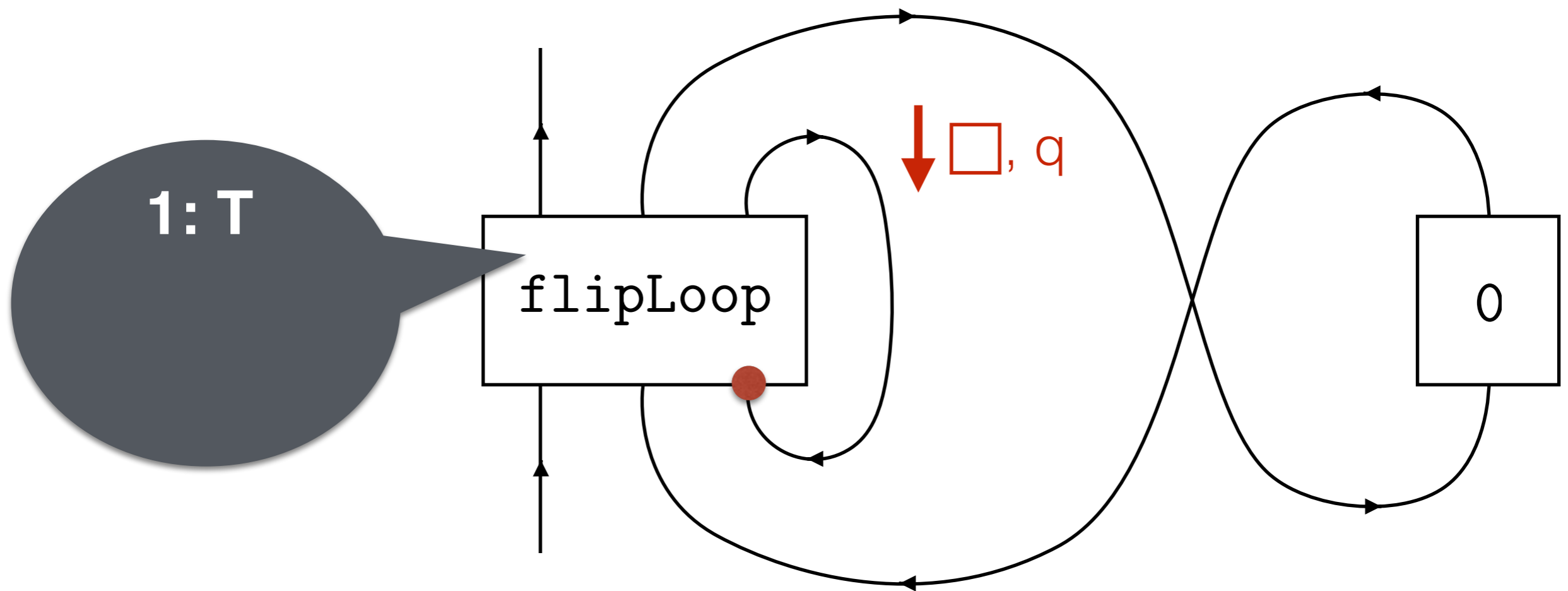
# Example: Recursive Probabilistic Program

```
(rec(flipLoop, x). choose0.4(x, flipLoop(x + 1))) 0
```



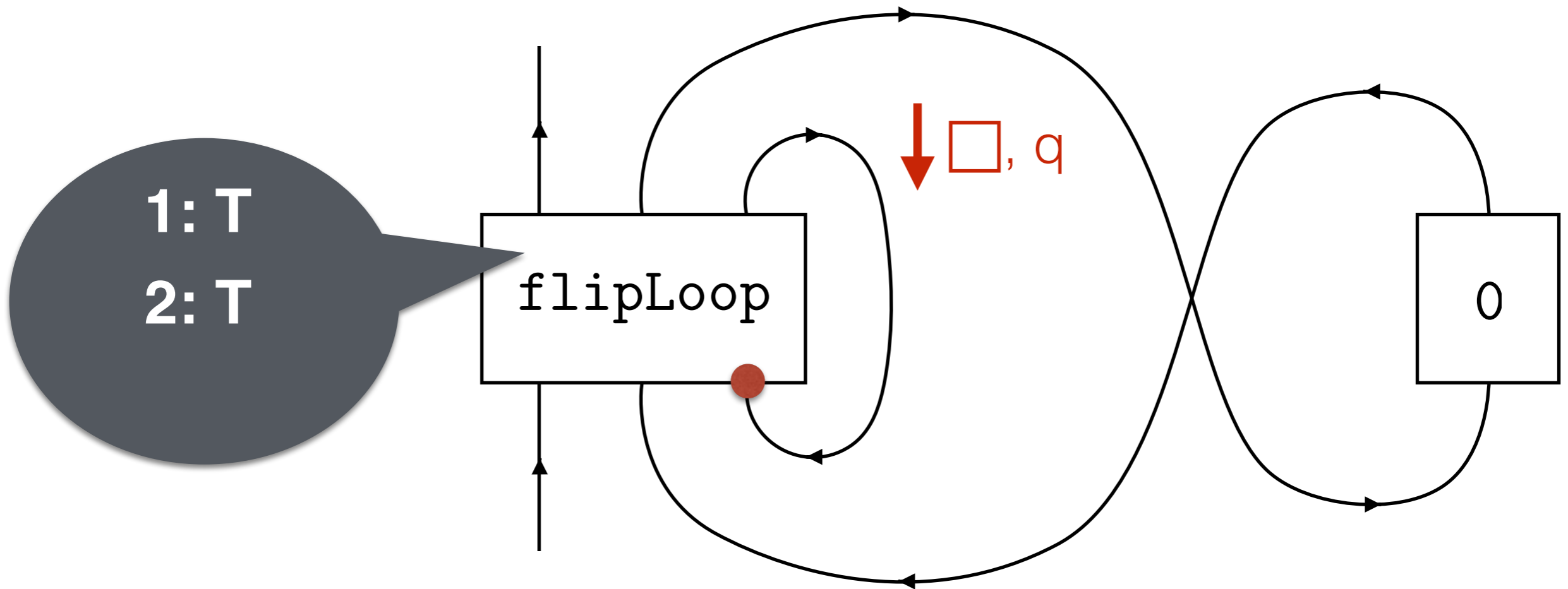
# Example: Recursive Probabilistic Program

```
(rec(flipLoop, x). choose0.4(x, flipLoop(x + 1))) 0
```



# Example: Recursive Probabilistic Program

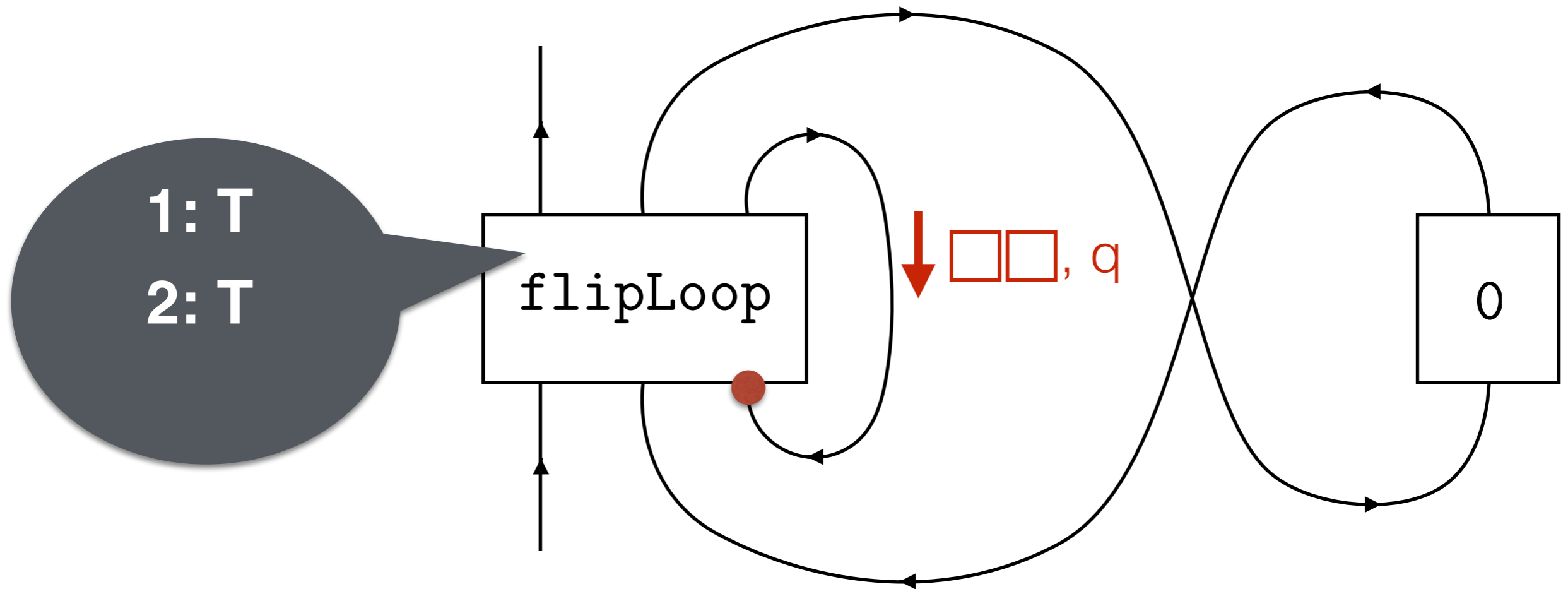
```
(rec(flipLoop, x). choose0.4(x, flipLoop(x + 1))) 0
```





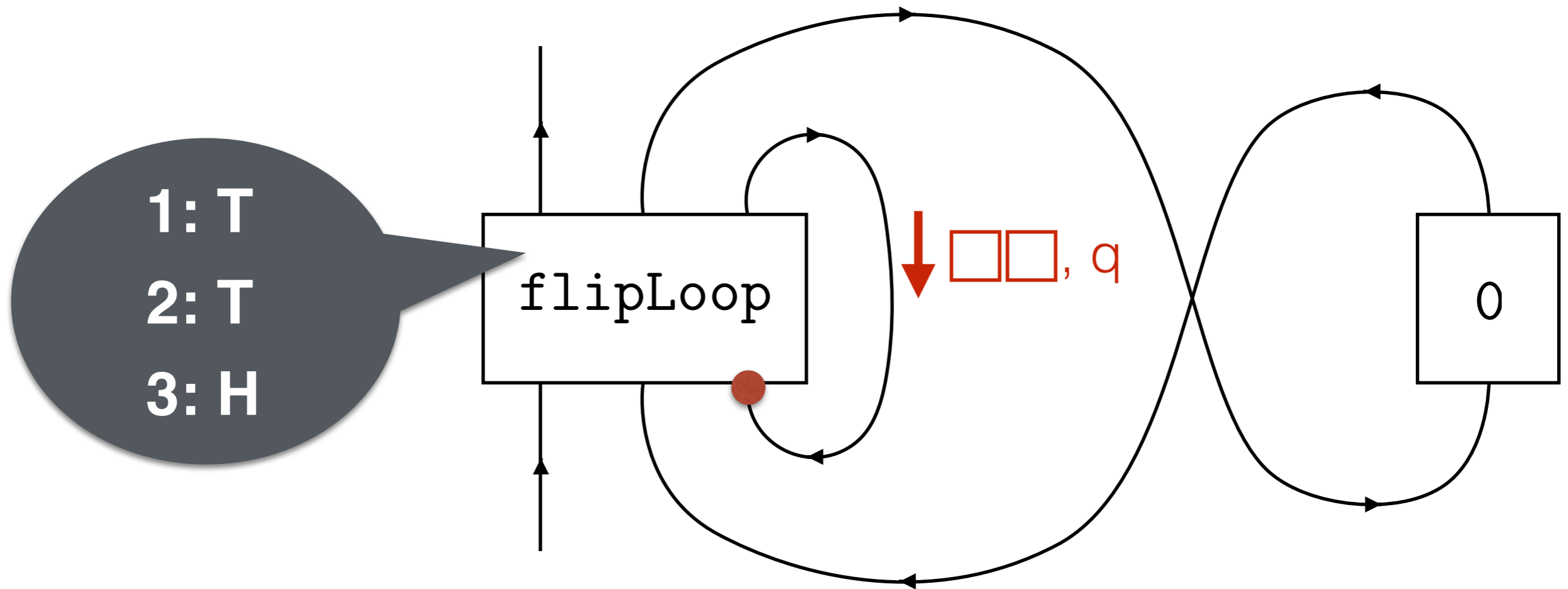
# Example: Recursive Probabilistic Program

```
(rec(flipLoop, x). choose0.4(x, flipLoop(x + 1))) 0
```



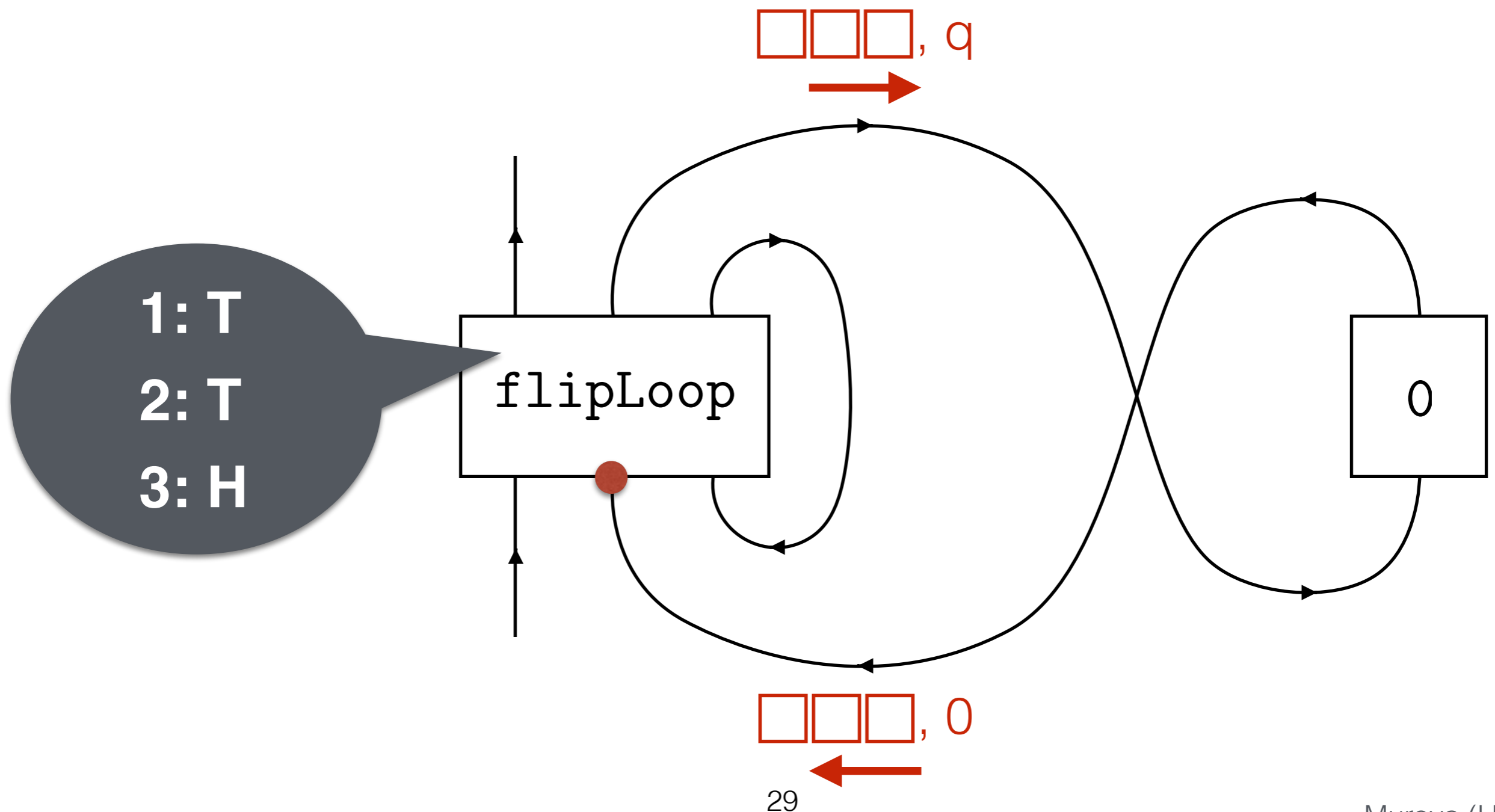
# Example: Recursive Probabilistic Program

```
(rec(flipLoop, x). choose0.4(x, flipLoop(x + 1))) 0
```



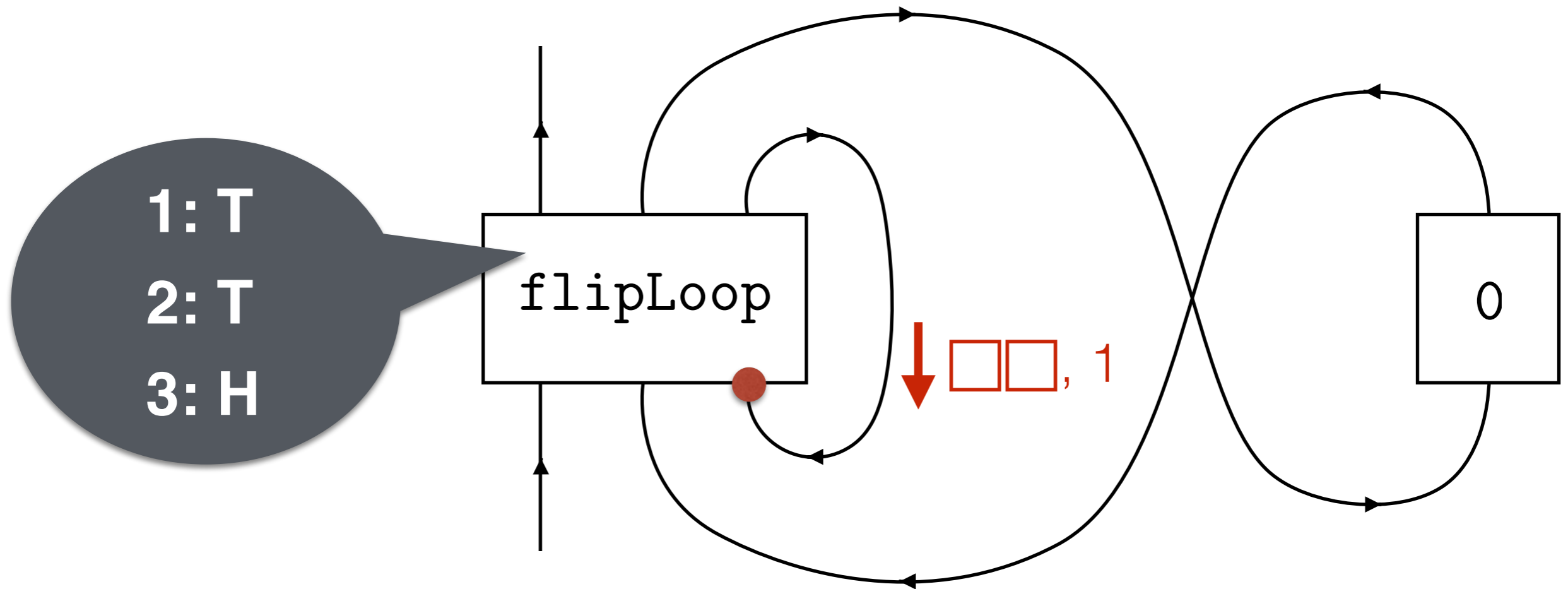
# Example: Recursive Probabilistic Program

```
(rec(flipLoop, x). choose0.4(x, flipLoop(x + 1))) 0
```



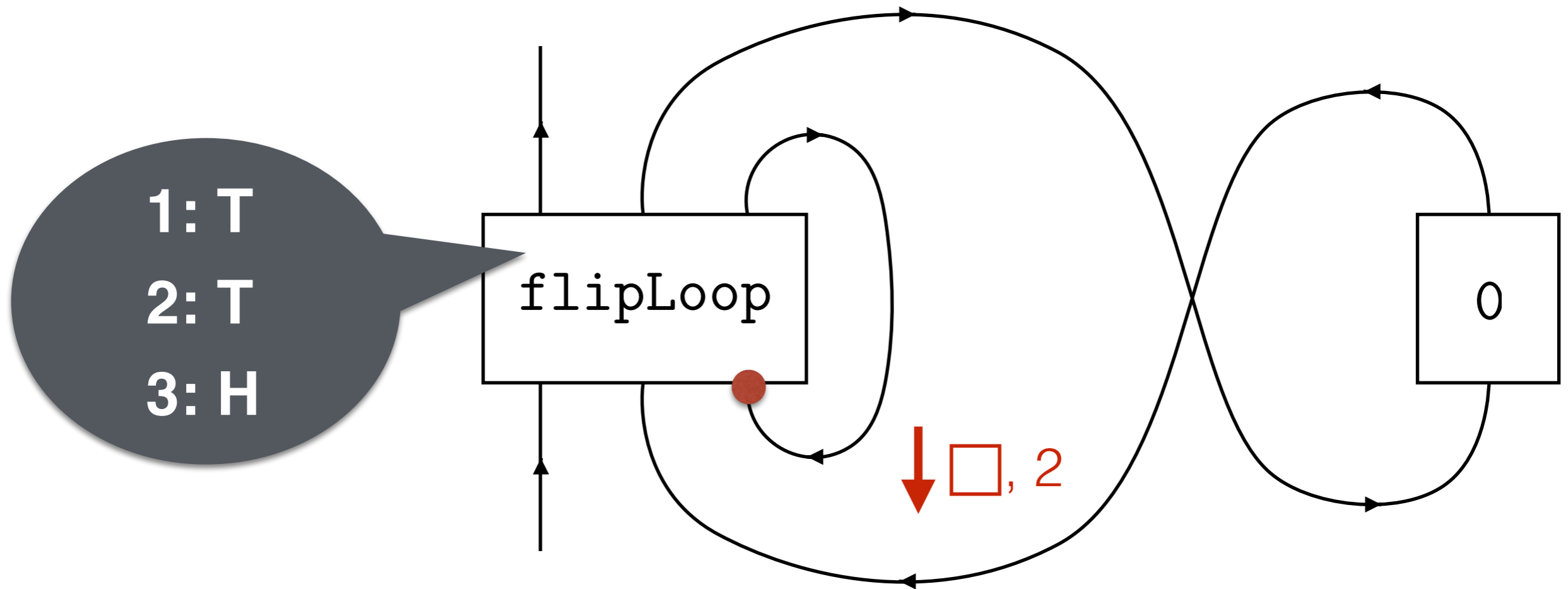
# Example: Recursive Probabilistic Program

```
(rec(flipLoop, x). choose0.4(x, flipLoop(x + 1))) 0
```



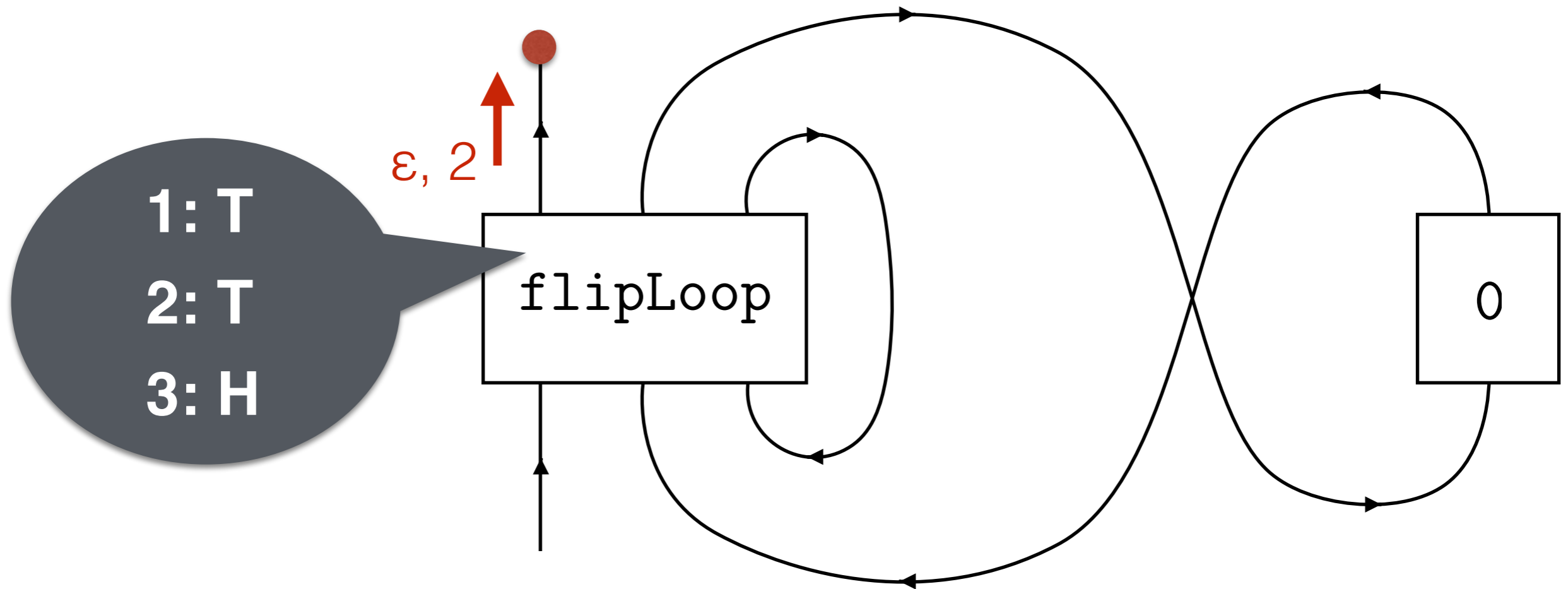
# Example: Recursive Probabilistic Program

```
(rec(flipLoop, x). choose0.4(x, flipLoop(x + 1))) 0
```



# Example: Recursive Probabilistic Program

```
(rec(flipLoop, x). choose0.4(x, flipLoop(x + 1))) 0
```



# Our Tool *TtT*

<http://koko-m.github.io/TtT/>

The screenshot shows a web browser window with the title "TtT" and the URL "koko-m.github.io/TtT/". The main content area has a dark red header with the text "TtT (Terms to Transducers)". Below the header, there is a text input field with the placeholder text "Enter a term, or type ";ex" to select one from 13 examples. [read documents]". The input field contains the code snippet `((rec(flipLoopSimple x) (choose(0.4) x (flipLoopSimple x))) 0)`. To the right of the input field is a control bar with play, pause, and next buttons, a progress slider, and a "300" value. The main content area is currently blank. On the right side of the browser window, there is a vertical sidebar with two empty rectangular boxes.

This is a simulation tool of the [memoryful Gol](#) framework.  
Implemented by [Koko Muroya](#), using [Processing.js](#) v1.4.8 and [PEG.js](#) v0.8.0.

# Our Tool *TtT*

<http://koko-m.github.io/TtT/>

The screenshot shows a web browser window with the title "TtT" and the URL "koko-m.github.io/TtT/". The main content area features a dark red header with the text "TtT (Terms to Transducers)". Below the header, there is a text input field containing the code snippet: `((rec(flipLoopSimple x) (choose(0.4) x (flipLoopSimple x))) 0)`. To the right of the input field is a control bar with play, pause, and next buttons, a progress slider, and a "300" value. The main simulation area is currently blank and greyed out. A footer at the bottom of the page provides context: "This is a simulation tool of the [memoryful Gol](#) framework. Implemented by [Koko Muroya](#), using [Processing.js](#) v1.4.8 and [PEG.js](#) v0.8.0."



# Memoryful GoI

<http://koko-m.github.io/TtT/>

**effectful  
PCF terms**



**transducers**

$$(\Gamma \vdash M : \tau) = \begin{array}{c} \begin{array}{c} N \uparrow \quad N \uparrow \\ \boxed{(\Gamma \vdash M : \tau)} \\ N \uparrow \quad N \uparrow \quad \dots \quad N \uparrow \end{array} \end{array}$$

adequate translation

via coalgebraic component calculus

