Cooking concurrency for algebraic effects

Mauro Jaskelioff
Universidad Nacional de Rosario, Argentina
(joint work with Exequiel Rivas)

Shonan Meeting 146, March 2019

Two Coin Tosses

$$\mathrm{toss} : 1 \to \mathbb{B}$$

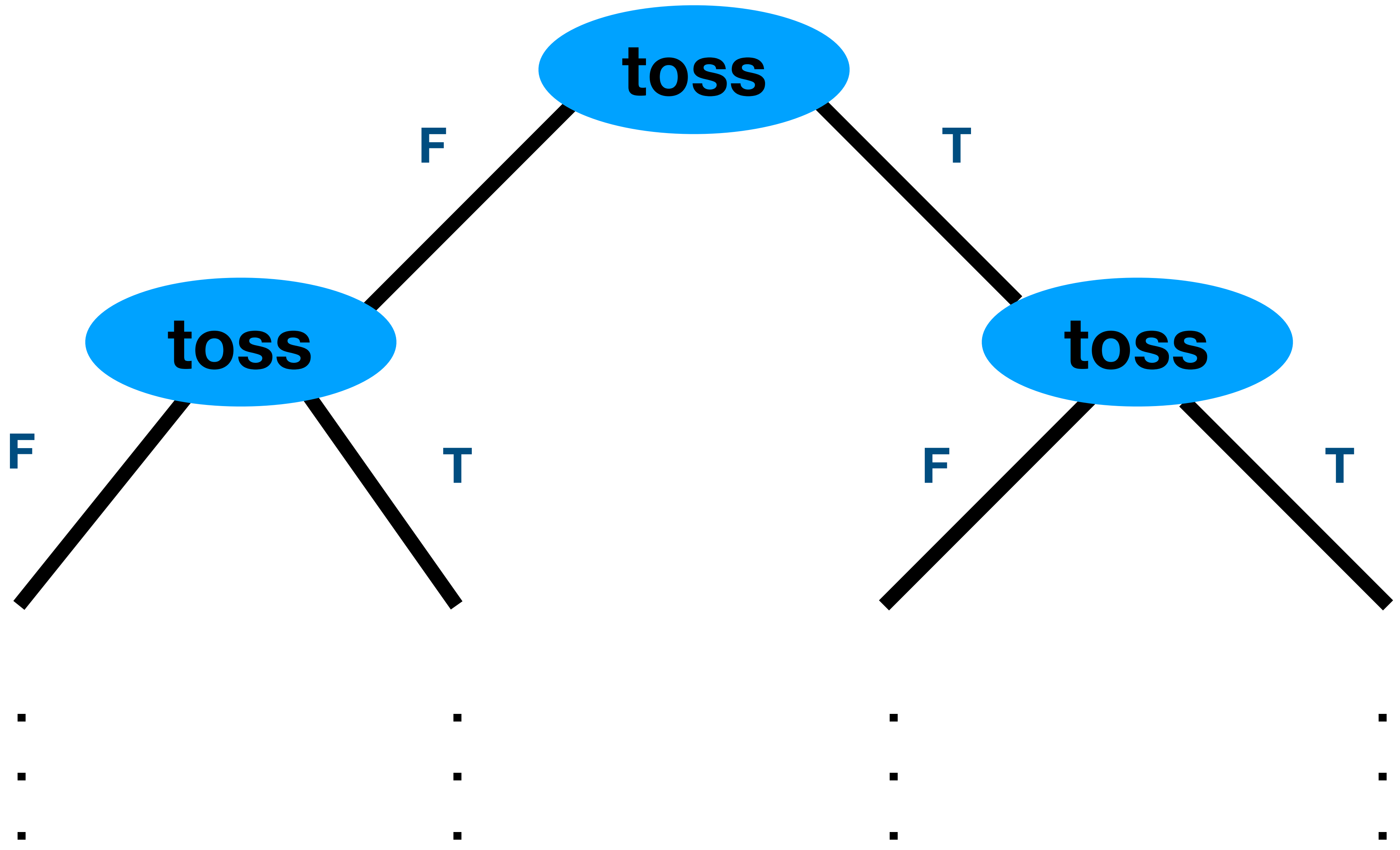
$$\begin{aligned}\mathbf{do}\ b_1 &\leftarrow \mathrm{toss} \\ b_2 &\leftarrow \mathrm{toss} \\ &\vdots\end{aligned}$$

Two Coin Tosses

$$\mathsf{toss} : 1 \to \mathbb{B}$$

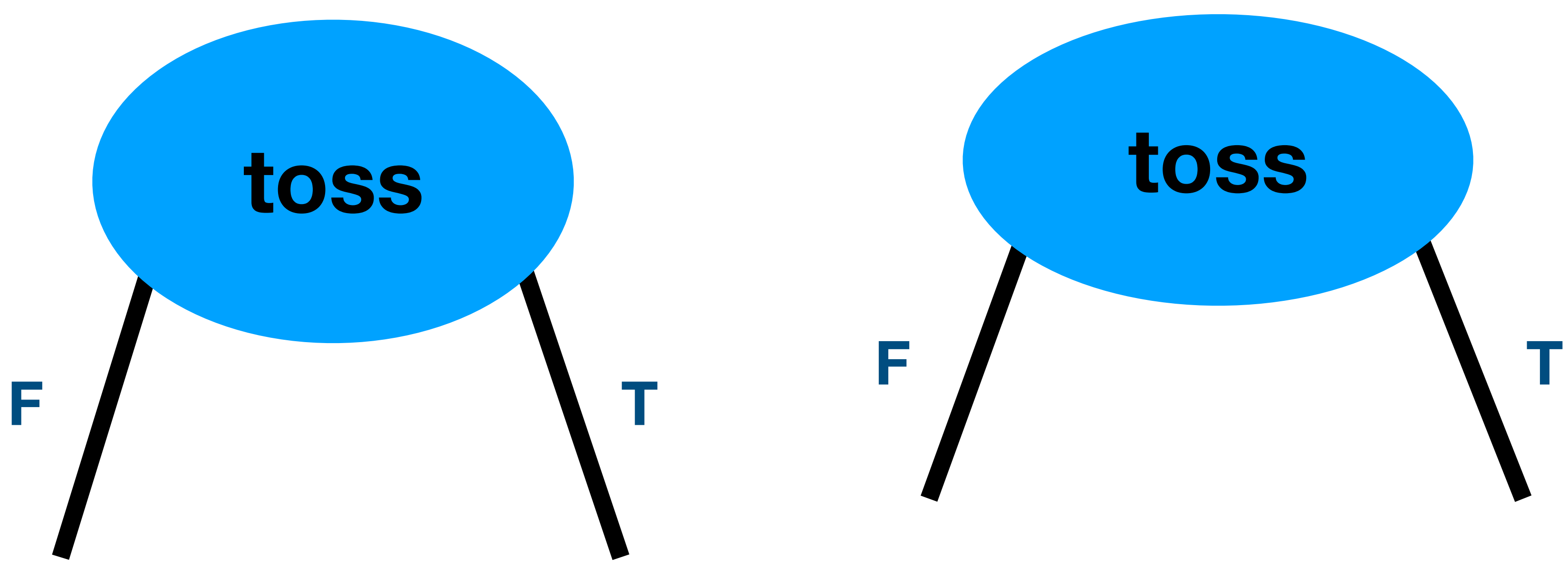
$$\begin{aligned} \mathbf{do}\ b_1 &\leftarrow \mathsf{toss} \\ b_2 &\leftarrow \mathsf{toss} \\ &\vdots \end{aligned}$$

Two Coin Tosses



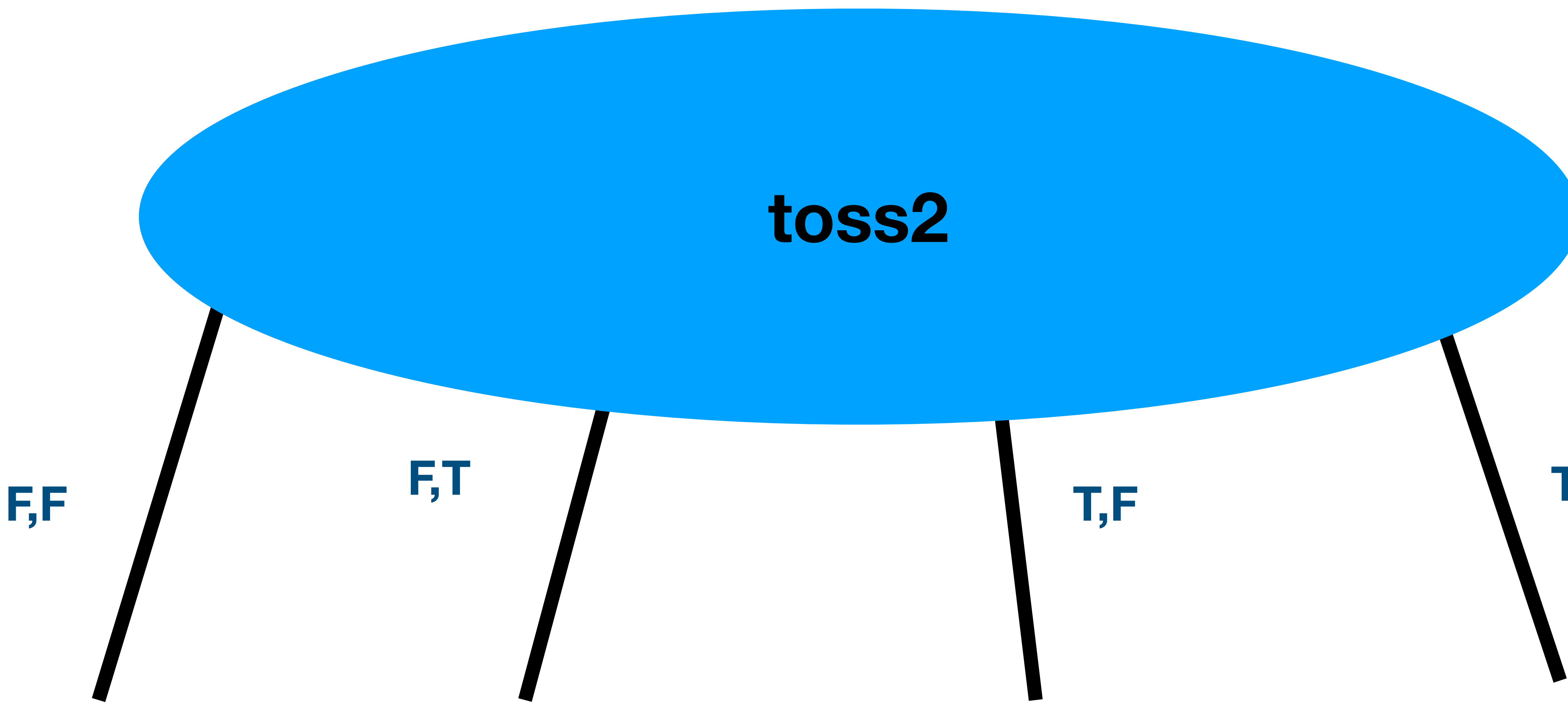
Two Independent Coin Tosses?

Two Independent Coin Tosses?



Toss Two Coins

$$\mathrm{toss2} : 1 \to \mathbb{B} \times \mathbb{B}$$



$$\text{toss2} : 1 \to \mathbb{B} \times \mathbb{B}$$

$$\text{toss2} : 1 \to \mathbb{B} \times \mathbb{B}$$

$$\text{toss3} : 1 \to \mathbb{B} \times \mathbb{B} \times \mathbb{B}$$

$$\mathrm{toss2} : 1 \to \mathbb{B} \times \mathbb{B}$$

$$\mathrm{toss3} : 1 \to \mathbb{B} \times \mathbb{B} \times \mathbb{B}$$

$$\mathrm{toss4} : 1 \to \mathbb{B} \times \mathbb{B} \times \mathbb{B} \times \mathbb{B}$$

$$\mathrm{toss5} : 1 \to \mathbb{B} \times \mathbb{B} \times \mathbb{B} \times \mathbb{B} \times \mathbb{B}$$

$$\mathrm{toss6} : 1 \to \mathbb{B} \times \mathbb{B} \times \mathbb{B} \times \mathbb{B} \times \mathbb{B} \times \mathbb{B}$$

$$\mathrm{toss7} : 1 \to \mathbb{B} \times \mathbb{B} \times \mathbb{B} \times \mathbb{B} \times \mathbb{B} \times \mathbb{B} \times \mathbb{B}$$

$$\mathrm{toss8} : 1 \to \mathbb{B} \times \mathbb{B} \times \mathbb{B} \times \mathbb{B} \times \mathbb{B} \times \mathbb{B} \times \mathbb{B} \times \mathbb{B}$$

$$\mathrm{toss9} : 1 \to \mathbb{B} \times \mathbb{B} \times \mathbb{B} \times \mathbb{B} \times \mathbb{B} \times \mathbb{B} \times \mathbb{B} \times \mathbb{B} \times \mathbb{B}$$

$$\mathrm{toss2} : 1 \to \mathbb{B} \times \mathbb{B}$$

$$\mathrm{toss3} : 1 \to \mathbb{B} \times \mathbb{B} \times \mathbb{B}$$

$$\mathrm{toss4} : 1 \to \mathbb{B} \times \mathbb{B} \times \mathbb{B} \times \mathbb{B}$$

$$\mathrm{toss5} : 1 \to \mathbb{B} \times \mathbb{B} \times \mathbb{B}$$

∞ Handler Clauses?

$$\times \mathbb{B} \times \mathbb{B}$$

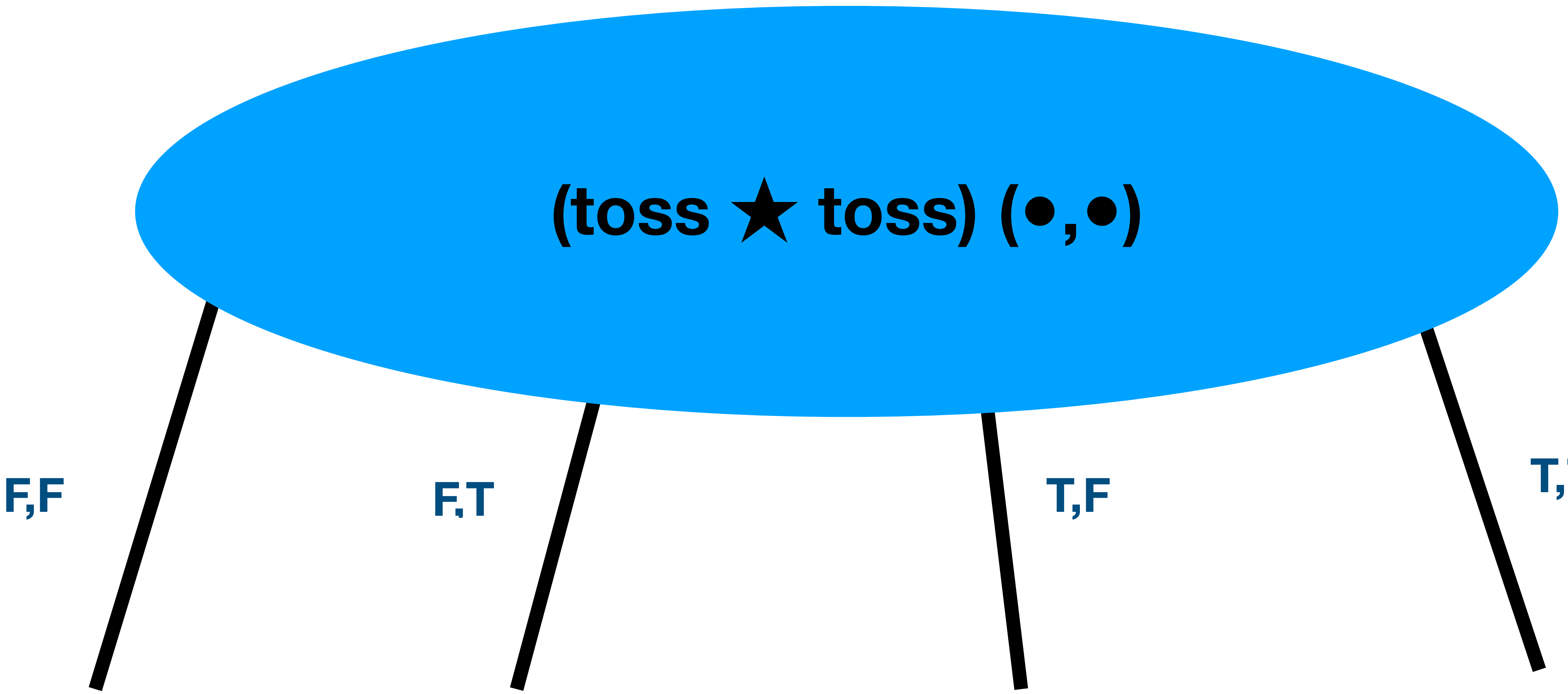
$$: 1 \to \mathbb{B} \times \mathbb{B} \times \mathbb{B} \times \mathbb{B} \times \mathbb{B} \times \mathbb{B} \times \mathbb{B}$$

$$\mathrm{toss8} : 1 \to \mathbb{B} \times \mathbb{B} \times \mathbb{B} \times \mathbb{B} \times \mathbb{B} \times \mathbb{B} \times \mathbb{B} \times \mathbb{B}$$

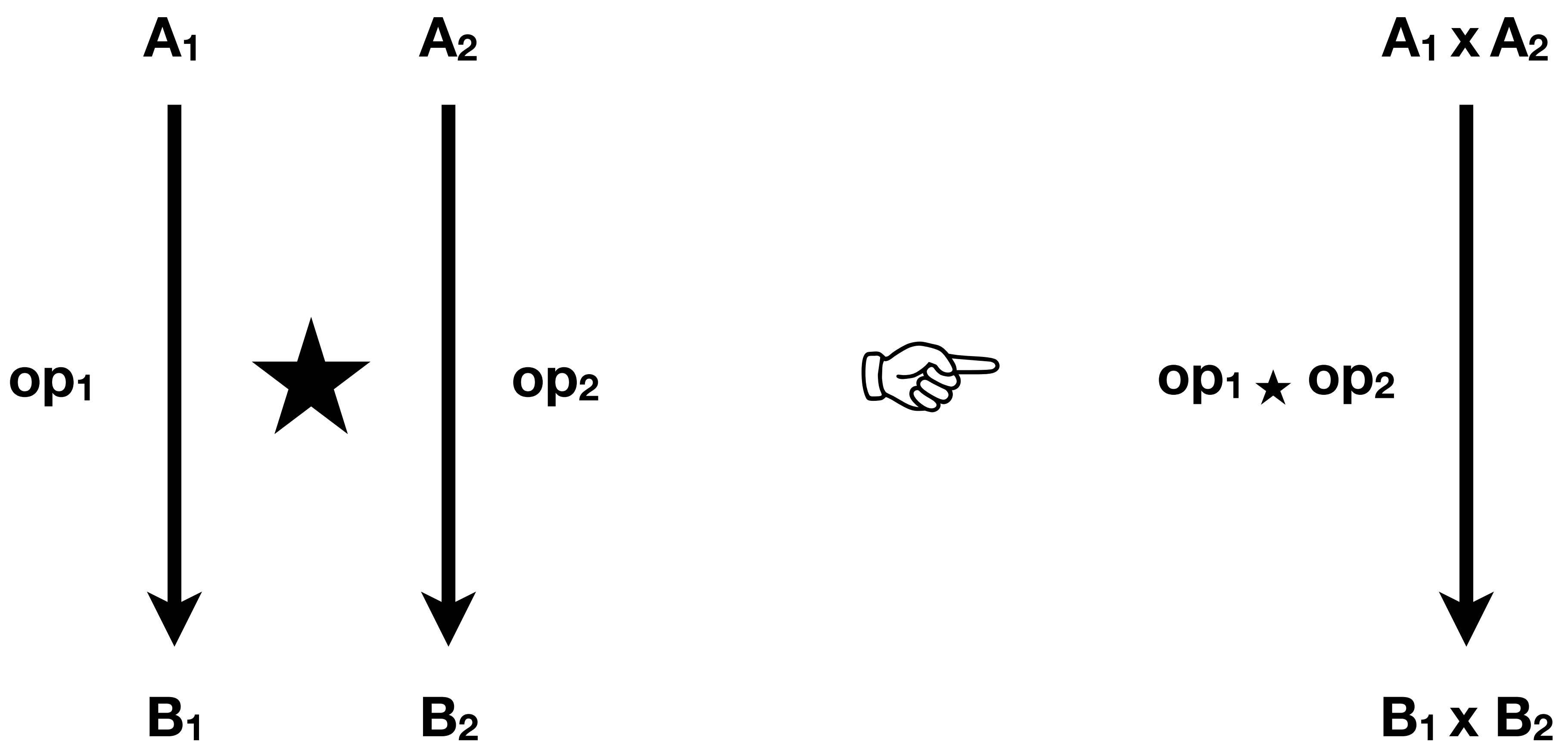
$$\mathrm{toss9} : 1 \to \mathbb{B} \times \mathbb{B} \times \mathbb{B} \times \mathbb{B} \times \mathbb{B} \times \mathbb{B} \times \mathbb{B} \times \mathbb{B} \times \mathbb{B}$$

Concurrent operations

$$\mathsf{toss} \star \mathsf{toss} : 1 \times 1 \to \mathbb{B} \times \mathbb{B}$$



Concurrent operations



Closure Signature

Closure Signature

$$\Sigma$$

Atomic operations

Closure Signature

$$\Sigma$$

Atomic operations

$$\Sigma^\star$$

Concurrent operations

Closure Signature

$$\Sigma \qquad\qquad \Sigma^{\star}$$

Atomic operations | **Concurrent operations**

$\{op_1 : A_1 \to B_1, op_2 : A_2 \to B_2\}$

Closure Signature

$$\Sigma$$

Atomic operations

$\{op_1 : A_1 \to B_1,\ op_2 : A_2 \to B_2\}$

$$\Sigma^{\star}$$

Concurrent operations

$\{op_1 : A_1 \to B_1,\ op_2 : A_2 \to B_2,$
$op_1{\star}op_1 : A_1{\times}A_1 \to B_1{\times}B_1,$
$op_1{\star}op_2 : A_1{\times}A_2 \to B_1{\times}B_2,$
$op_2{\star}op_1 : A_2{\times}A_1 \to B_2{\times}B_1,$
$op_1{\star}op_1{\star}op_1 : A_1{\times}A_1{\times}A_1 \to B_1{\times}B_1{\times}B_1,$
$op_1{\star}op_1{\star}op_2 : A_1{\times}A_1{\times}A_2 \to B_1{\times}B_1{\times}B_2,$
$\ldots\}$

An Eff Calculus (with products)

based on M. Pretnar's tutorial

value type $\quad A, B ::= \mathsf{bool} \mid \mathsf{unit}$

$\mid \quad A \to \underline{C}$

$\mid \quad \underline{C} \Rightarrow \underline{D}$

$\mid \quad A \times B$

computation type $\quad \underline{C}, \underline{D} ::= A \,!\, \{\mathsf{op}_i : A_i \to B_i\}_{1 \leqslant i \leqslant n}$

$\mathsf{op}_i \in \Sigma$

An Eff Calculus (with products)

based on M. Pretnar's tutorial

$$
\begin{array}{llcl}
\text{value} & v & ::= & x \\
& & \mid & \mathbf{true} \mid \mathbf{false} \\
& & \mid & (v, v) \mid \bullet \\
& & \mid & \mathbf{fun}\ x \mapsto c \\
& & \mid & h
\end{array}
$$

$$
\begin{array}{lll}
\text{handler} & h ::= \mathbf{handler}\ \{ & \mathbf{return}\ x \mapsto c_r, \quad \} \\
& & \mathsf{op}_1(x;k) \mapsto c_1, \ldots, \mathsf{op}_n(x;k) \mapsto c_n
\end{array}
$$

An Eff Calculus (with products)

based on M. Pretnar's tutorial

$$
\begin{aligned}
\text{computation} \quad c ::= {} & \mathbf{return}\ v \\
\mid {} & \mathsf{op}(v; y \mapsto c) \\
\mid {} & \mathbf{do}\ x \leftarrow c_1\ \mathbf{in}\ c_2 \\
\mid {} & \mathbf{if}\ v\ \mathbf{then}\ c_1\ \mathbf{else}\ c_2 \\
\mid {} & v_1\ v_2 \\
\mid {} & \mathbf{with}\ v\ \mathbf{handle}\ c \\
\mid {} & \mathbf{match}\ v\ \mathbf{as}\ (x, y) \mapsto c
\end{aligned}
$$

An Eff Calculus (with products)

based on M. Pretnar's tutorial

$$
\begin{array}{lrl}
\text{computation} & c ::= & \textbf{return } v \\
& | & \mathsf{op}(v; y \mapsto c) \\
& | & \textbf{do } x \leftarrow c_1 \textbf{ in } c_2 \\
& | & \textbf{if } v \textbf{ then } c_1 \textbf{ else } c_2 \\
& | & v_1\ v_2 \\
& | & \textbf{with } v \textbf{ handle } c \\
& | & \boxed{\textbf{match } v \textbf{ as } (x, y) \mapsto c}
\end{array}
$$

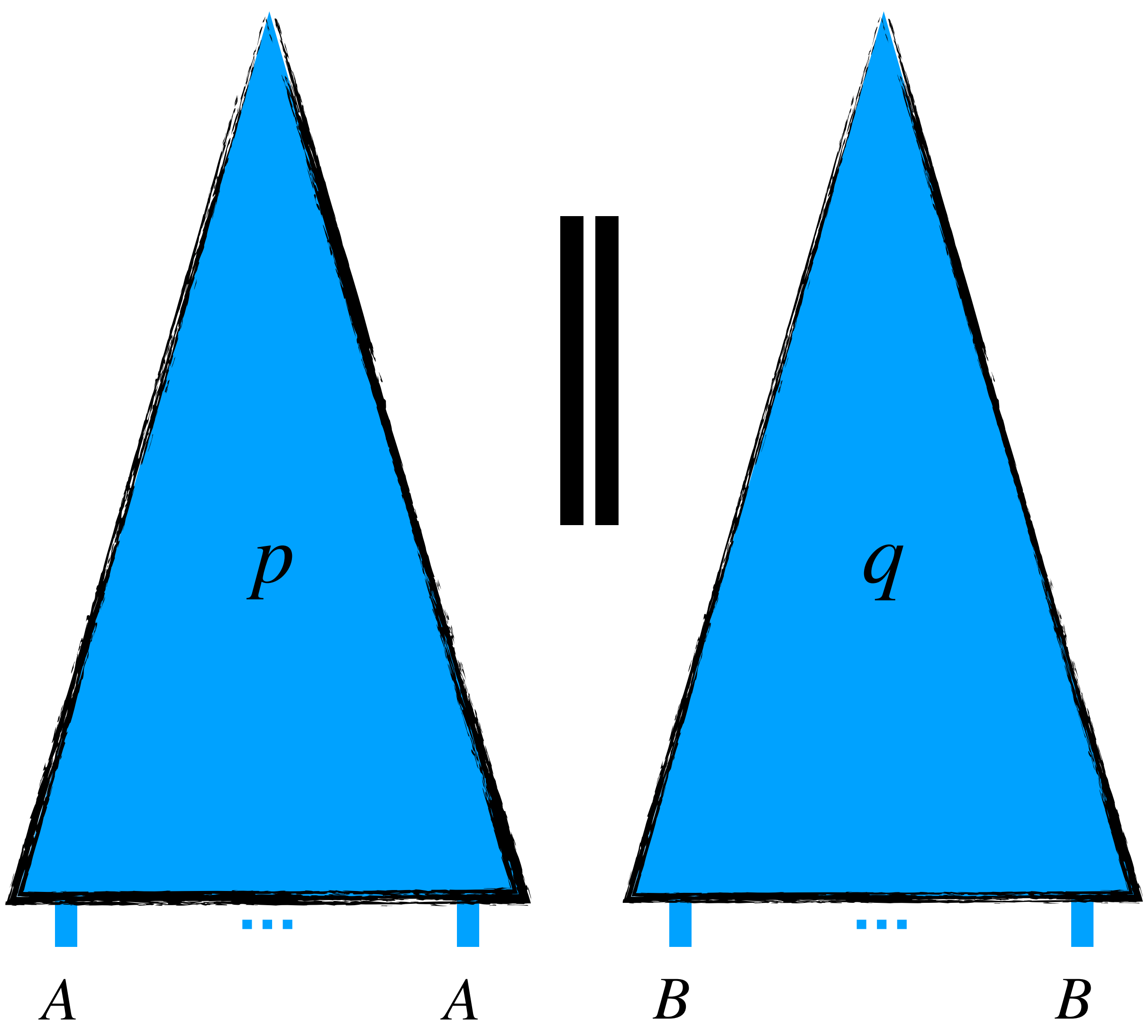
$\mathsf{op} \in \Sigma$

ChEff Calculus

$$
\begin{array}{lrl}
\text{computation} & c ::= & \mathbf{return}\ v \\
 & \mid & \mathsf{op}(v; y \mapsto c) \\
 & \mid & \mathbf{do}\ x \leftarrow c_1\ \mathbf{in}\ c_2 \\
 & \mid & \mathbf{if}\ v\ \mathbf{then}\ c_1\ \mathbf{else}\ c_2 \\
 & \mid & v_1\ v_2 \\
 & \mid & \mathbf{with}\ v\ \mathbf{handle}\ c \\
 & \mid & \mathbf{match}\ v\ \mathbf{as}\ (x, y) \mapsto c \\
 & \mid & c_1 \parallel c_2
\end{array}
$$

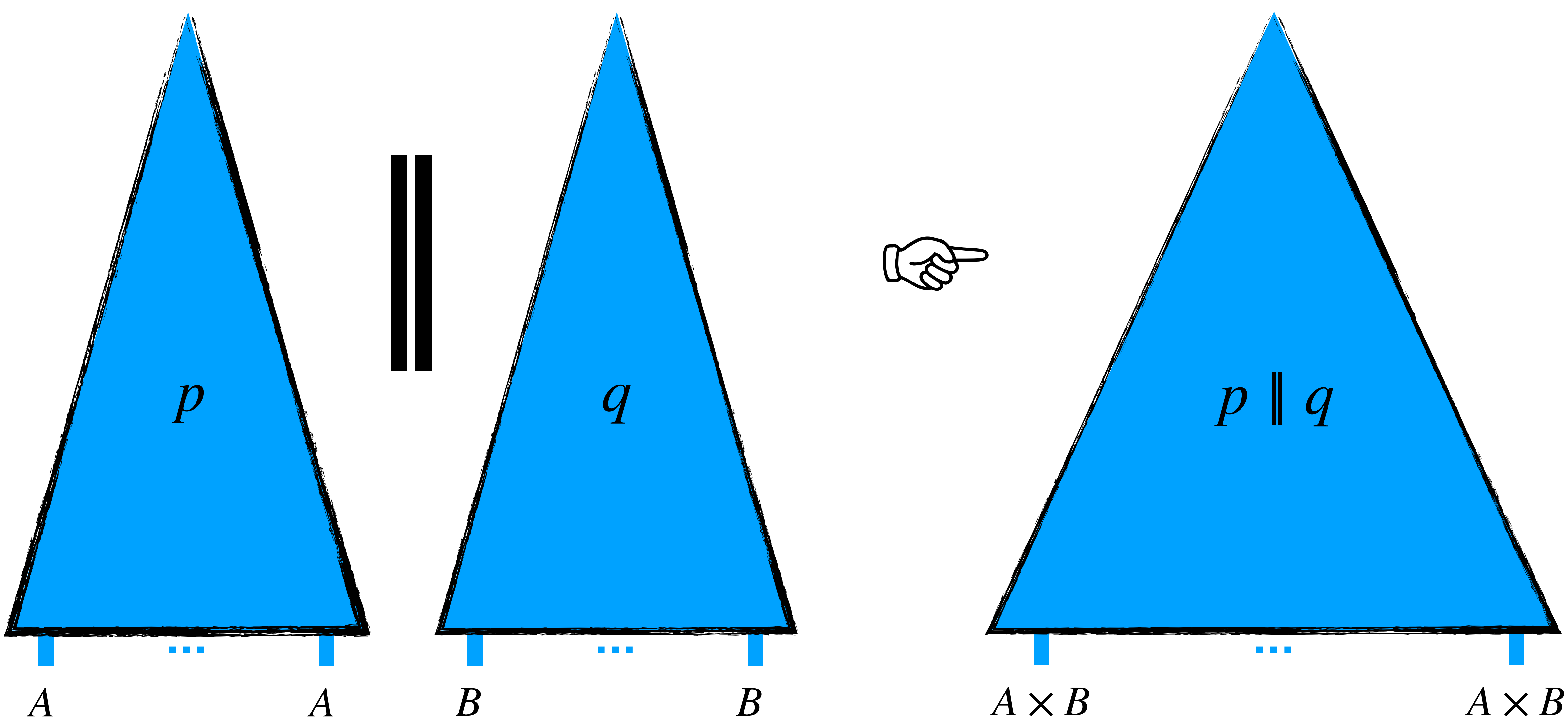
Merging Computations

$p : A ! \Delta$ $q : B ! \Delta$

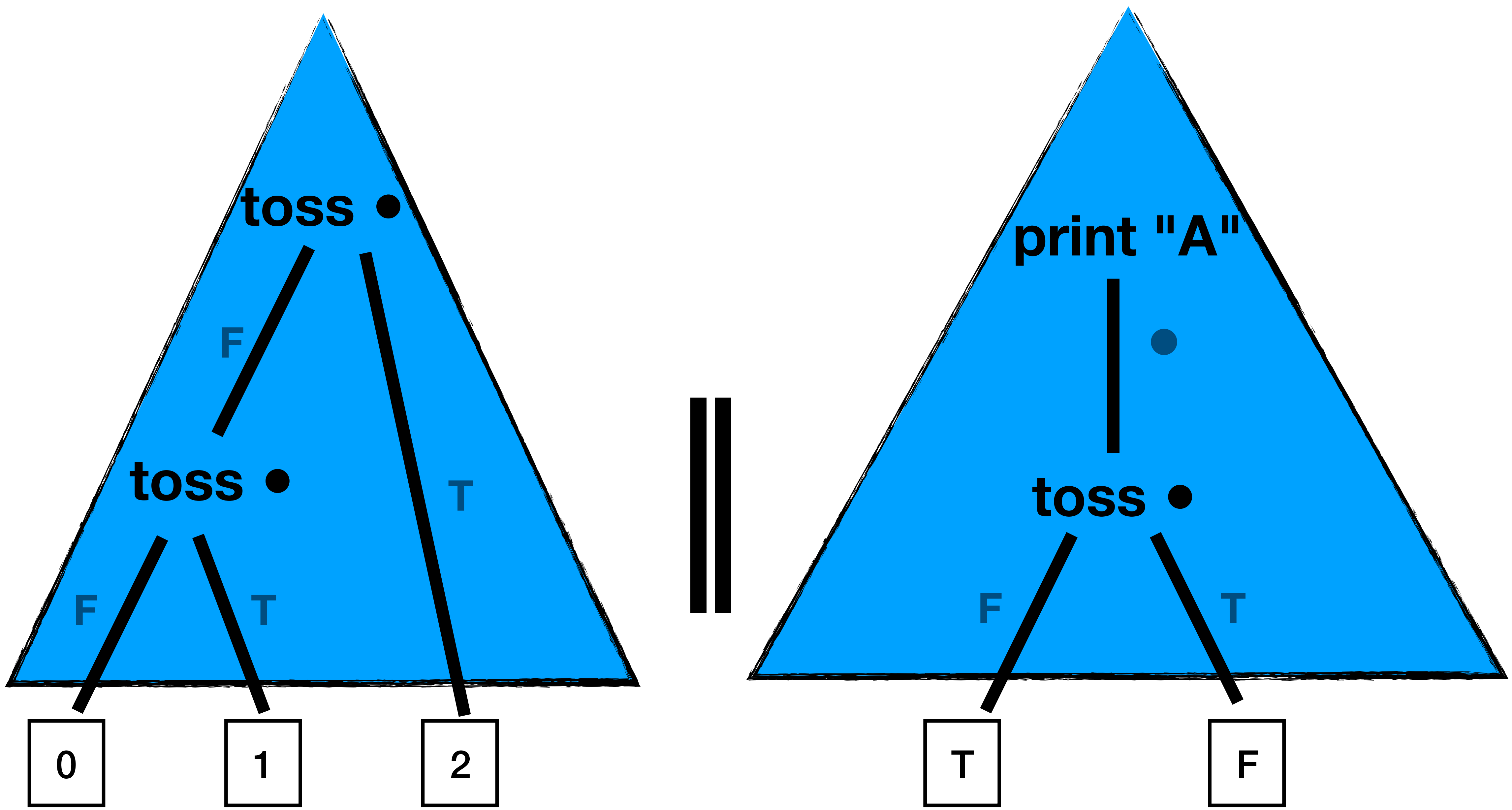


Merging Computations

$p : A ! \Delta$ $\quad q : B ! \Delta$ $\quad p \parallel q : A \times B \,!\, \Delta$



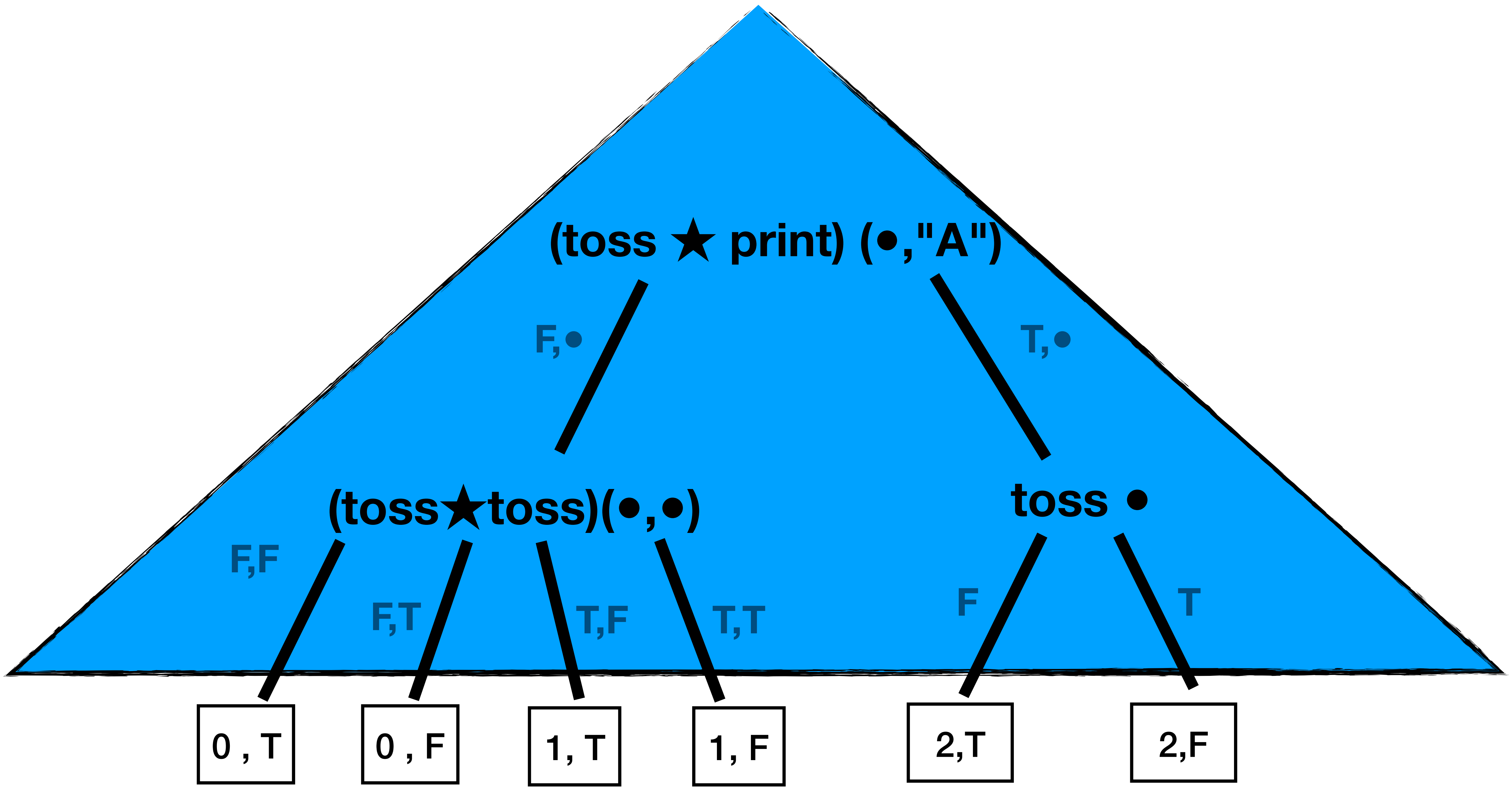
Merging Computations



$\mathbb{N}$ **! {toss, print}**

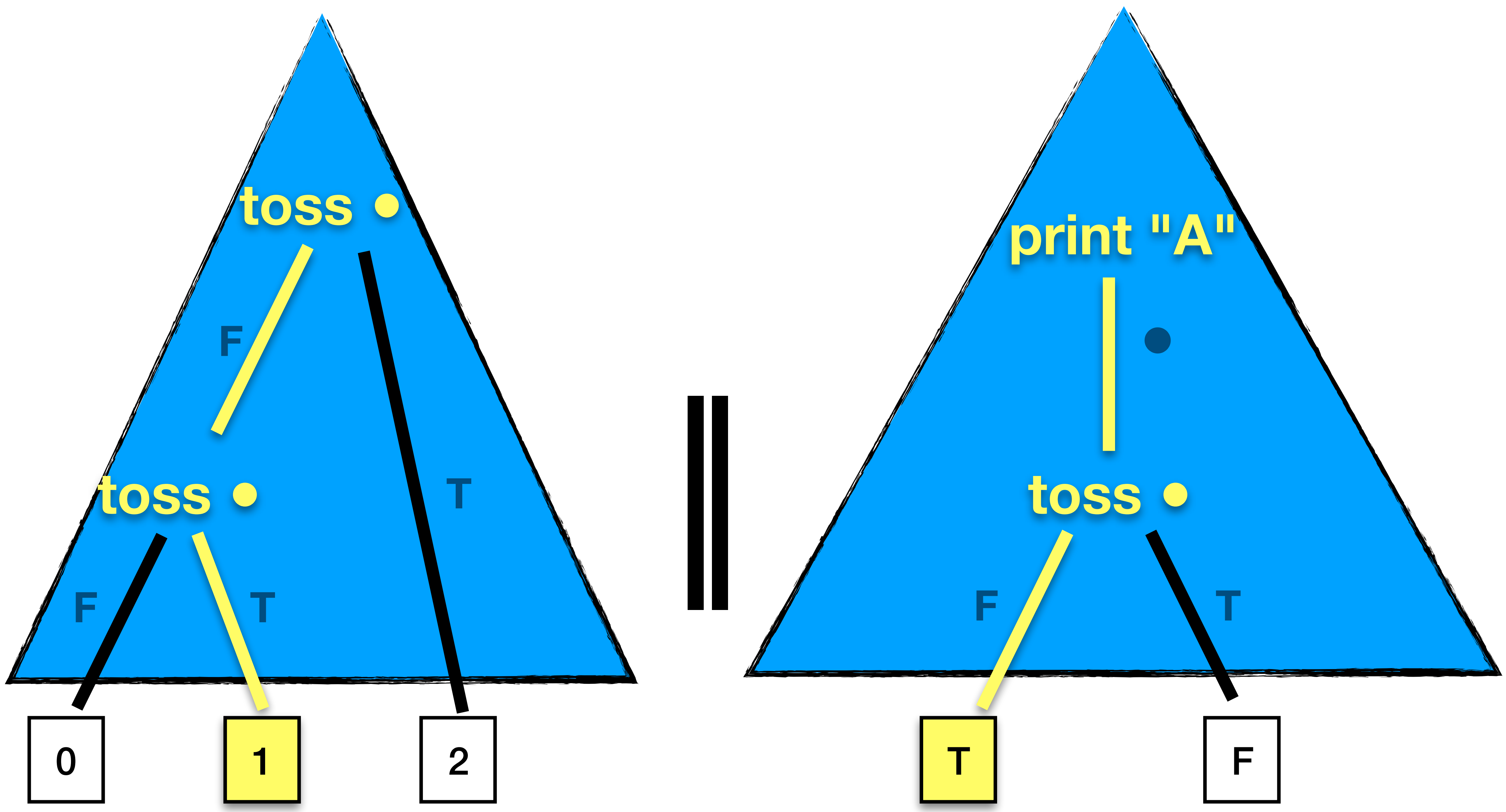
$\mathbb{B}$ **! {toss, print}**

Merging Computations

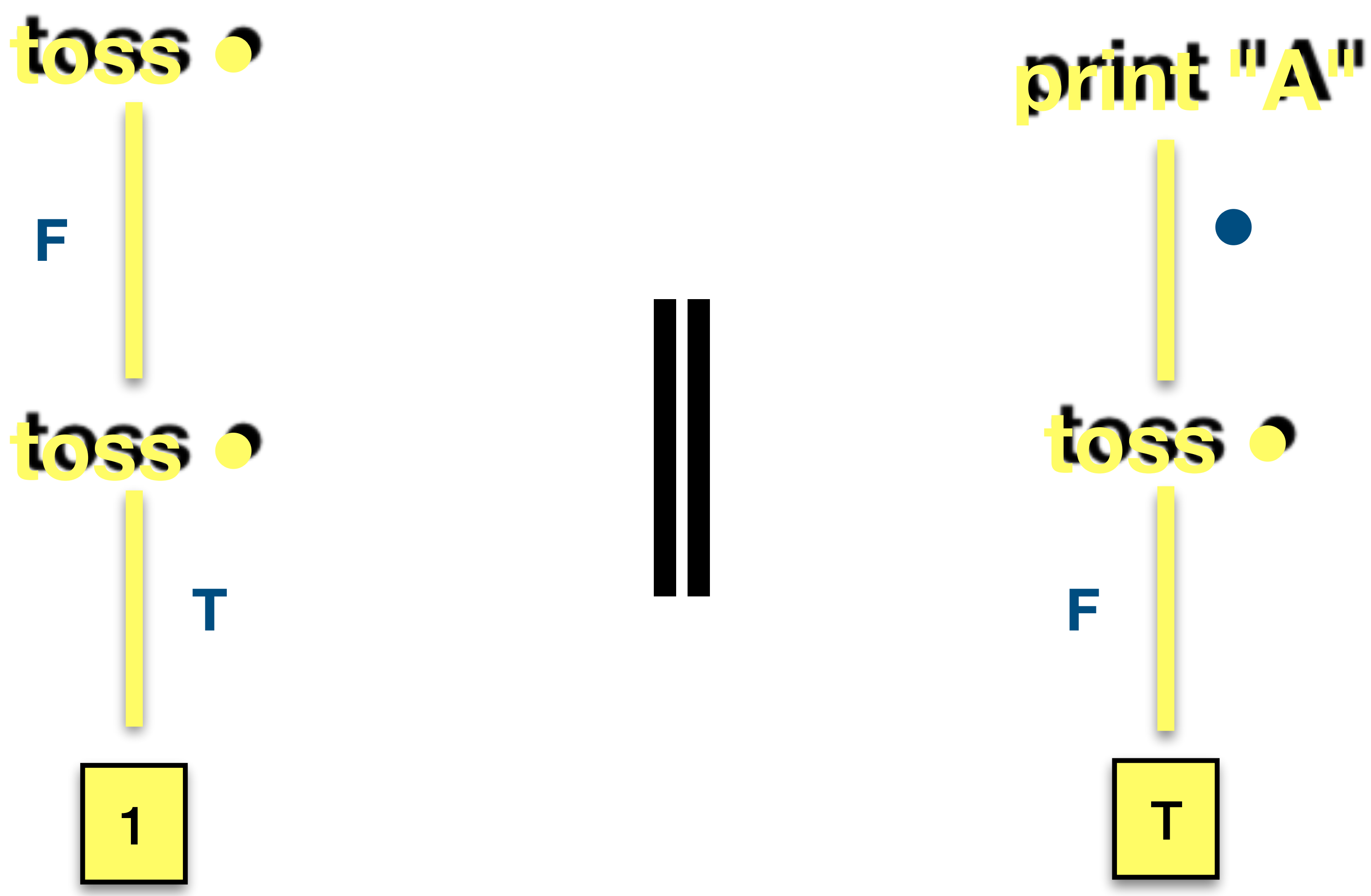


$\mathbb{N}\times\mathbb{B}$! {toss, print}

Merging Computations

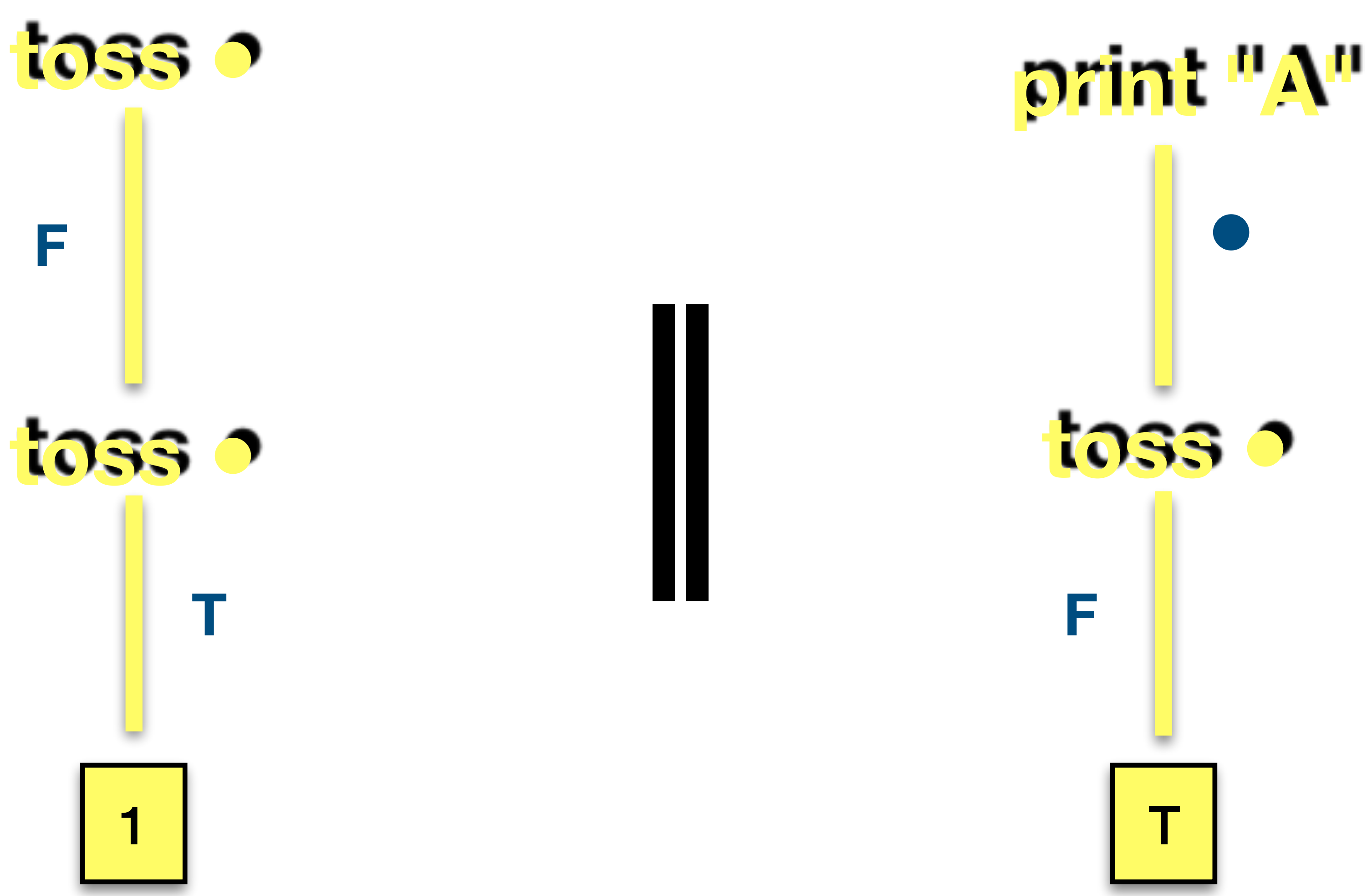


Merging Computations



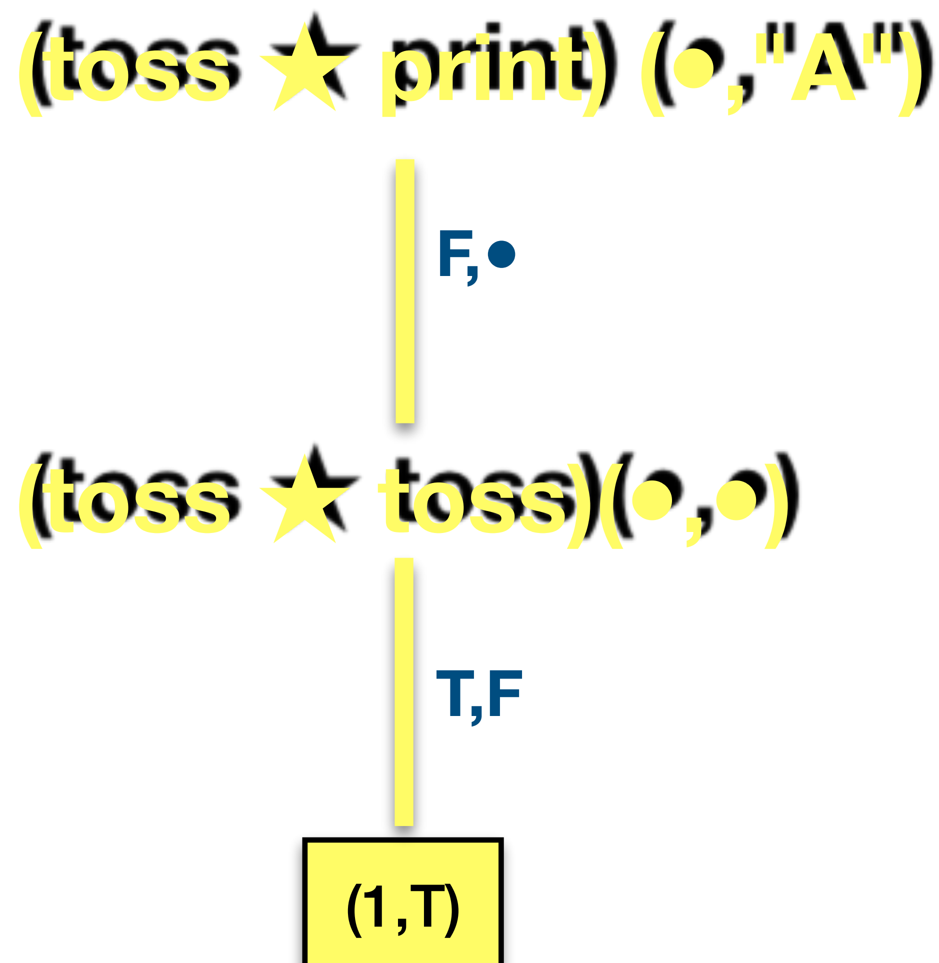
Merging Computations

is zipping



Merging Computations

is zipping



Handling Concurrent Computations

Handling Concurrent Computations

- Reuse usual handlers

HANDLER

$$\frac{\Gamma, x : A \vdash_{\mathsf{c}} c_r : B \,!\, \Delta' \qquad \big[(\mathsf{op}_i : A_i \to B_i) \in \Delta \quad \Gamma, x : A_i, k : B_i \to B \,!\, \Delta' \vdash_{\mathsf{c}} c_i : B \,!\, \Delta'\big]_{1 \leqslant i \leqslant n} \qquad \Delta / \{\mathsf{op}_i\}_{1 \leqslant i \leqslant n} \subseteq \Delta'}{\Gamma \vdash \textbf{handler}\ \{\textbf{return}\ x \mapsto c_r, \mathsf{op}_1(x;k) \mapsto c_1, \ldots, \mathsf{op}_n(x;k) \mapsto c_n\} : A \,!\, \Delta \Rightarrow B \,!\, \Delta'}$$

- Preserve semantics when || is not used

Handling Concurrent Computations

- Reuse usual handlers

HANDLER

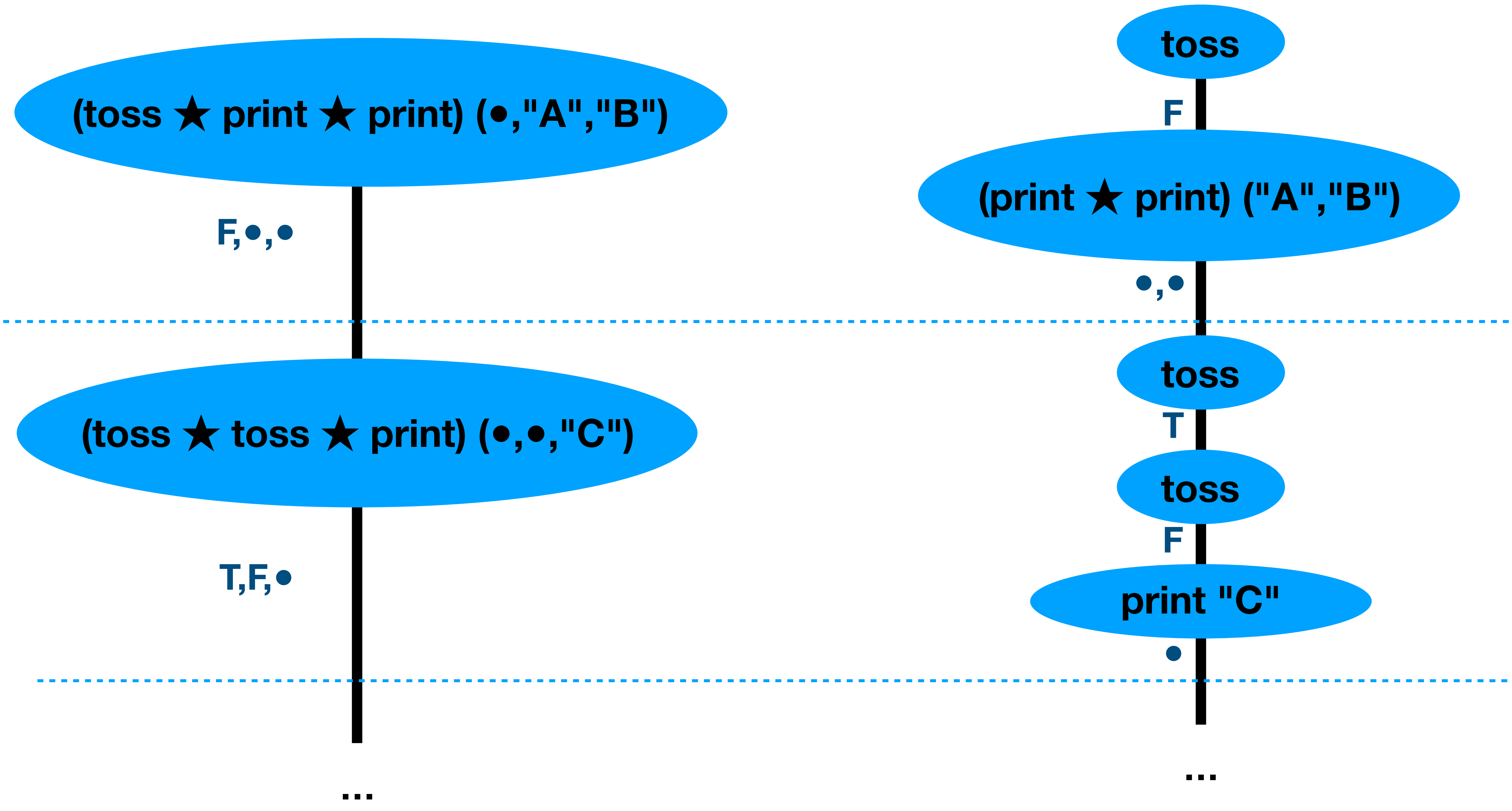
$$\frac{\Gamma, x : A \vdash_{\mathsf{c}} c_r : B \mathbin{!} \Delta' \quad \left[(\mathsf{op}_i : A_i \to B_i) \in \Delta \quad \Gamma, x : A_i, k : B_i \to B \mathbin{!} \Delta' \vdash_{\mathsf{c}} c_i : B \mathbin{!} \Delta'\right]_{1 \leqslant i \leqslant n} \qquad \Delta / \{\mathsf{op}_i\}_{1 \leqslant i \leqslant n} \subseteq \Delta'}{\Gamma \vdash \textbf{handler}\ \{\textbf{return}\ x \mapsto c_r, \mathsf{op}_1(x;k) \mapsto c_1, \ldots, \mathsf{op}_n(x;k) \mapsto c_n\} : A \mathbin{!} \Delta \Rightarrow B \mathbin{!} \Delta'}$$

- Preserve semantics when || is not used

- What happens when || is used?

Interleaving Semantics:

Handling toss



Interleaving vs Native concurrency

Interleaving vs Native concurrency

- Handling all operations we get an Eff program with an interleaving semantics

Interleaving vs Native concurrency

- Handling all operations we get an Eff program with an interleaving semantics

- Unhandled concurrent operations may given a native concurrency semantics

Interleaving vs Native concurrency

- Handling all operations we get an Eff program with an interleaving semantics
- Unhandled concurrent operations may given a native concurrency semantics
- Example: Haskell's IO monad has an operation

$$\text{concurrently} :: \text{IO } a \to \text{IO } b \to \text{IO } (a, b)$$

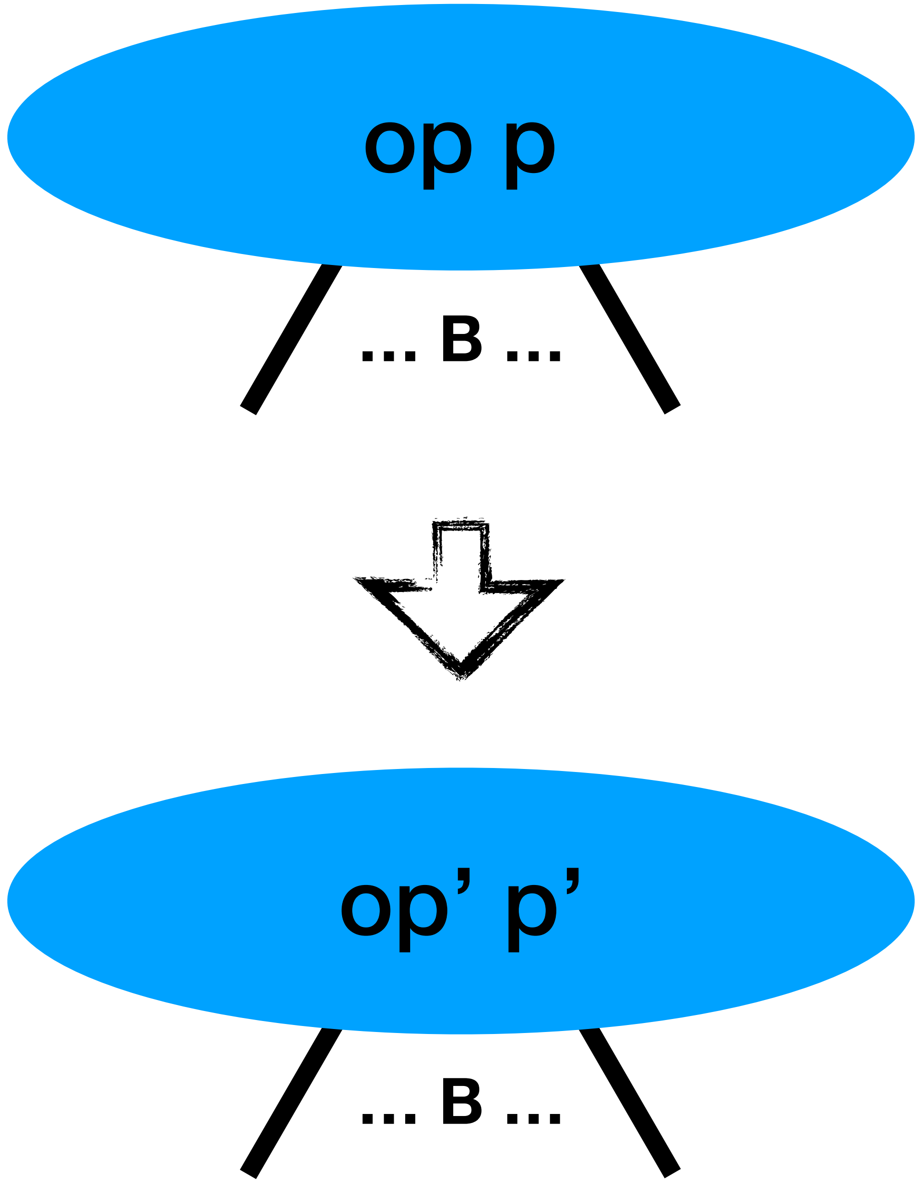
Concurrent Handlers

Concurrent Handlers

- Can’t use the continuation

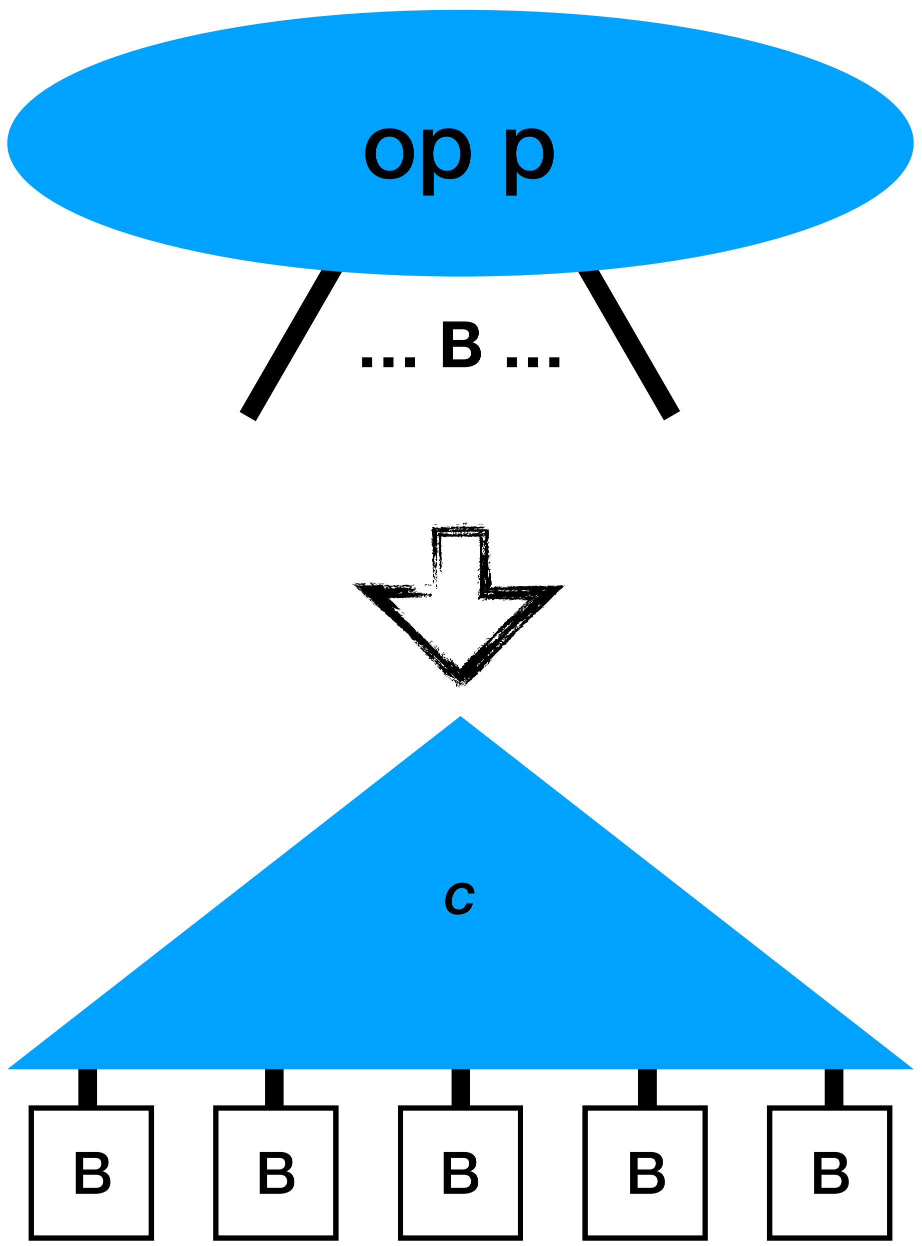
Concurrent Handlers

- Can't use the continuation
- Modify a concurrent node but preserve arity.



Concurrent Handlers

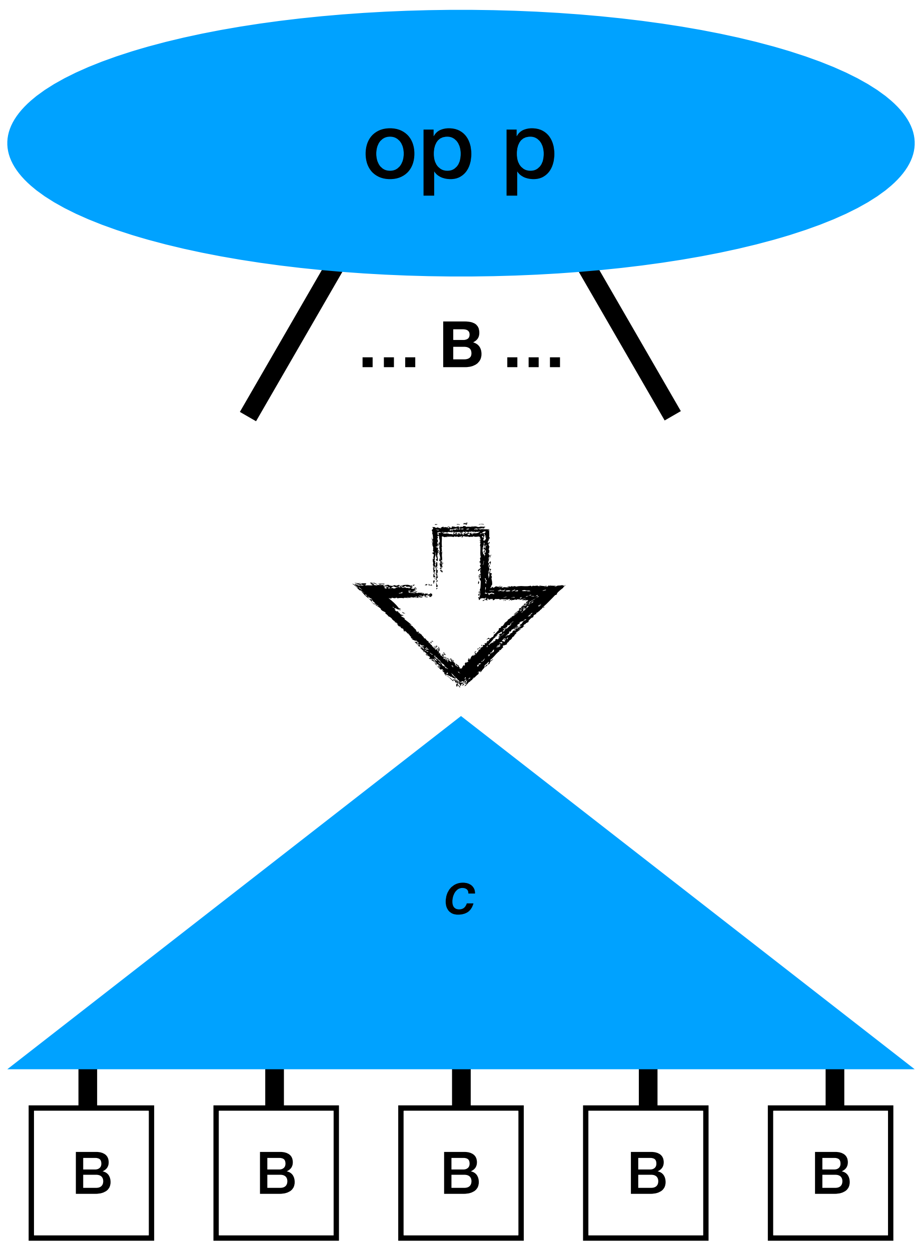
- Can’t use the continuation
- Modify a concurrent node but preserve arity.



Concurrent Handlers

- Can’t use the continuation
- Modify a concurrent node but preserve arity.
- In ChEff, for each op to be handled, we need a computation:

$$\Gamma, x\colon A_{\mathrm{op}}, \ldots \vdash_c c\colon B_{\mathrm{op}}!\Delta$$



Scenario

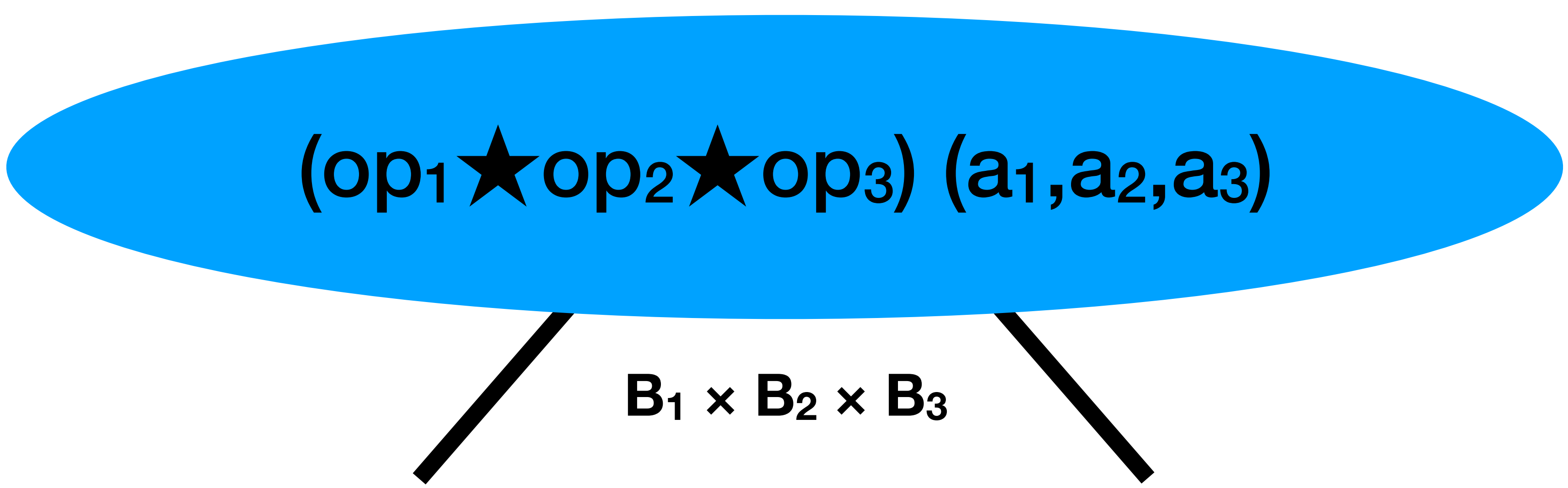
- We want to interpret an operation random : $1 \to$ int32, in terms of toss : $1 \to 2$

Concurrent Handler: **interpret**

$\mathsf{op}_1 : A_1 \to B_1$ $\mathsf{op}_2 : A_2 \to B_2$ $\mathsf{op}_3 : A_3 \to B_3$

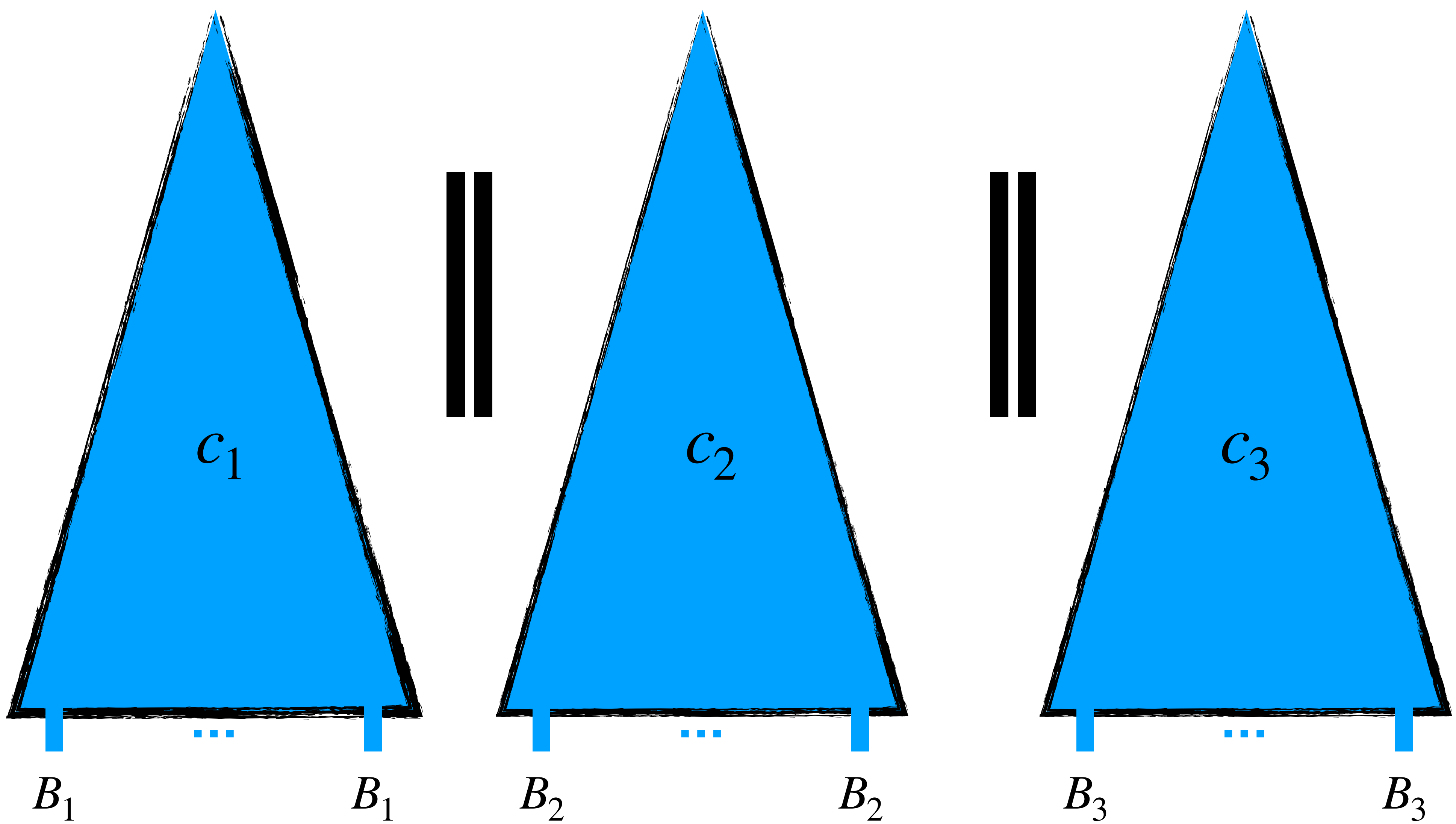
Concurrent Handler: interpret

$\mathsf{op}_1 : A_1 \to B_1 \qquad \mathsf{op}_2 : A_2 \to B_2 \qquad \mathsf{op}_3 : A_3 \to B_3$



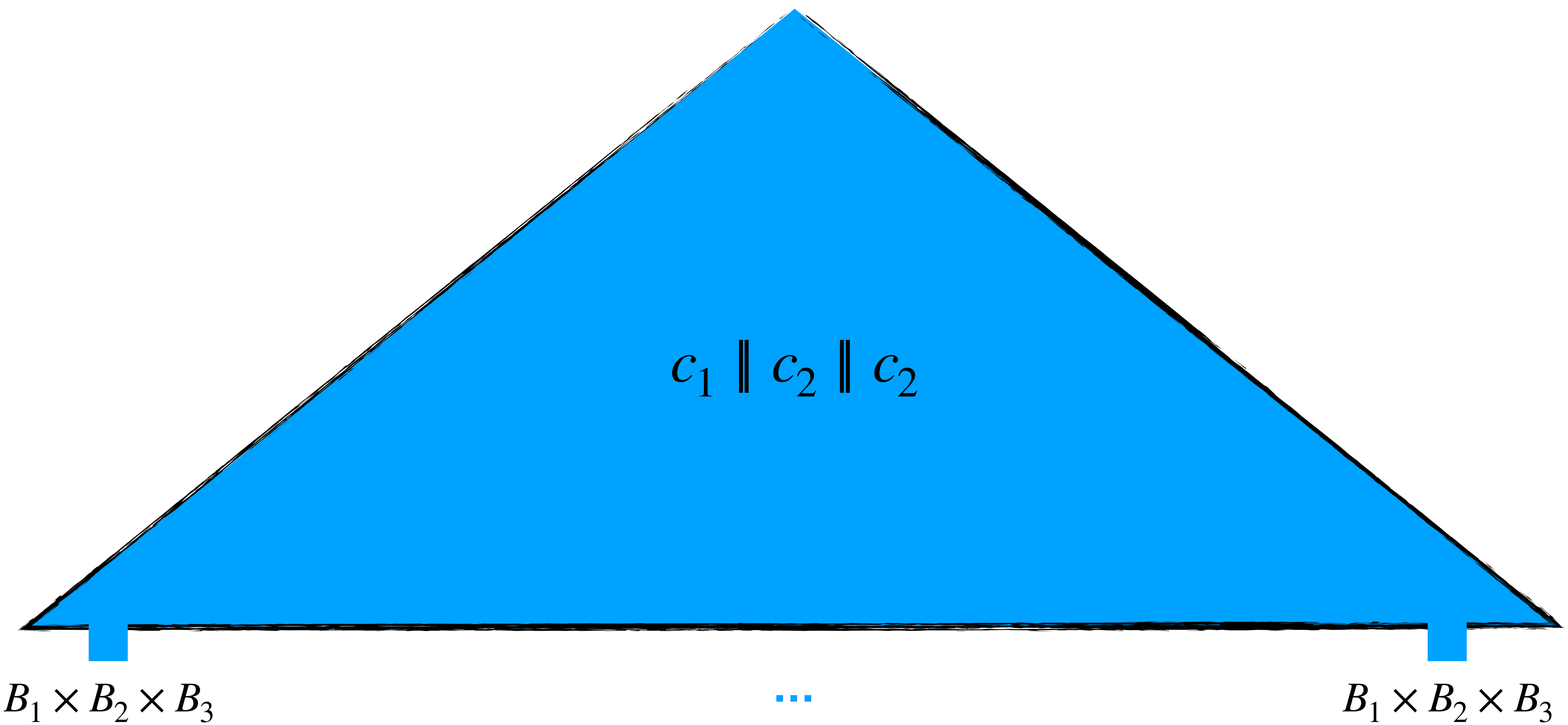
Concurrent Handler: **interpret**

$\mathsf{op}_1 : A_1 \to B_1$ $\qquad$ $\mathsf{op}_2 : A_2 \to B_2$ $\qquad$ $\mathsf{op}_3 : A_3 \to B_3$



Concurrent Handler: interpret

$\mathsf{op}_1 : A_1 \to B_1$ $\mathsf{op}_2 : A_2 \to B_2$ $\mathsf{op}_3 : A_3 \to B_3$



Concurrent Handler: interpret

- We add a new handler to ChEff:

$$
\begin{array}{lll}
\text{handler} & h ::= & \ldots \\
& \mid & \mathbf{interpret}_A \ \{ \ \mathsf{op}_1(x) \mapsto c_1, \ldots, \mathsf{op}_n(x) \mapsto c_n \ \}
\end{array}
$$

$$
\textsc{Interpret}\quad
\frac{\left[(\mathsf{op}_i : A_i \to B_i) \in \Delta \quad \Gamma, x : A_i \vdash_{\mathsf{c}} c_i : B_i \,!\, \Delta'\right]_{1 \leqslant i \leqslant n} \qquad \Delta / \{\mathsf{op}_i\}_{1 \leqslant i \leqslant n} \subseteq \Delta'}{\Gamma \vdash \mathbf{interpret}_A \ \{\mathsf{op}_1(x) \mapsto c_1, \ldots, \mathsf{op}_n(x) \mapsto c_n\} : A \,!\, \Delta \Rightarrow A \,!\, \Delta'}
$$

- The handler is parametric in its return type A

Scenario

- We want to consolidate concurrent queries to a database
- We need to:
  1. collect the queries
  2. produce a new computation based on the collected info
  3. The computation returns a big table with all the info. We need projections into the original queries.

Concurrent Handler: consolidate

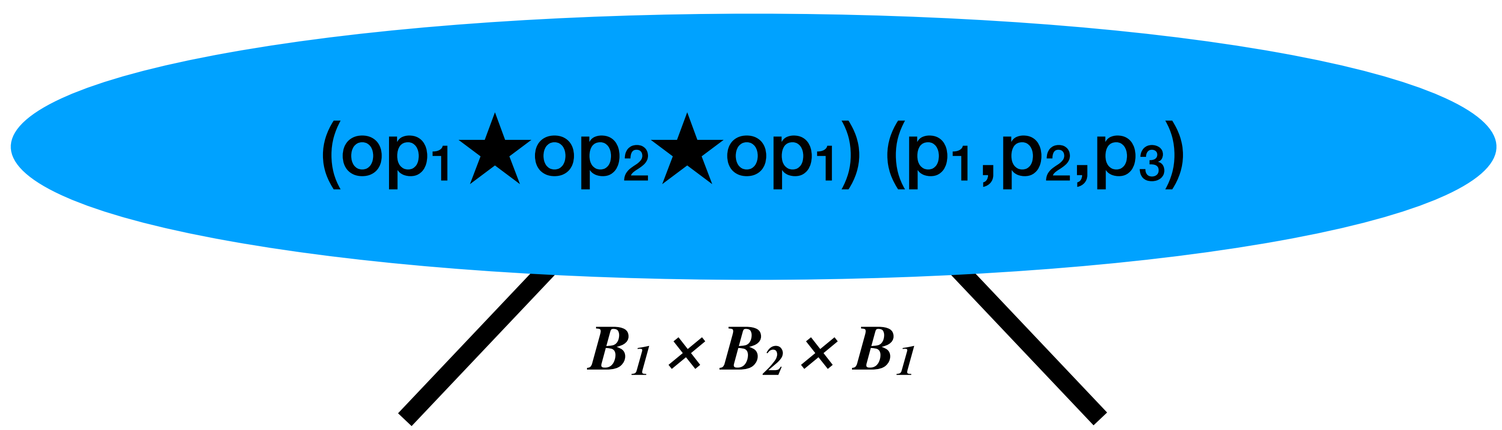
- handler $\quad h ::= \ldots$

$$\mid \quad \mathbf{consolidate}_A \left\{ \begin{array}{l} \mathsf{op}_1((x,r) \mapsto c_1; (x,y) \mapsto k_1), \\ \ldots, \\ \mathsf{op}_n((x,r) \mapsto c_n; (x,y) \mapsto k_n) \end{array} \right\} \mathbf{from}\ v\ \mathbf{with}\ c$$

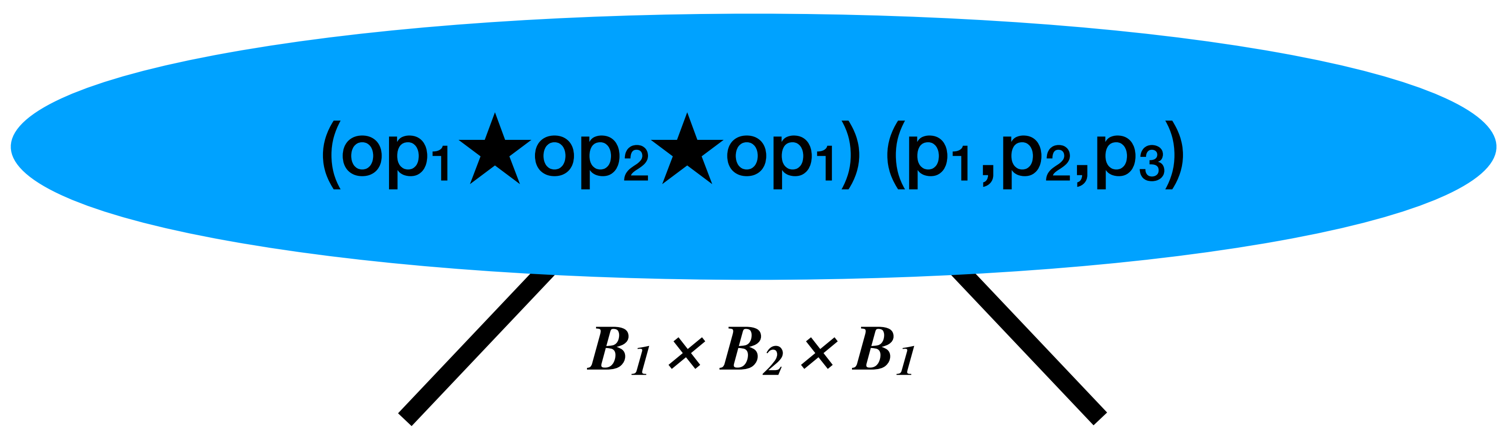
$$\textsc{Consolidate}\quad \frac{\begin{array}{c}\big[(\mathsf{op}_i : A_i \to B_i) \in \Delta \quad \Gamma, x : A_i, r : B \vdash_{\mathsf{c}} c_i : B \,!\, \Delta' \quad \Gamma, x : A_i, y : B' \vdash_{\mathsf{c}} k_i : B_i \,!\, \Delta'\big]_{1 \leqslant i \leqslant n} \\ \Gamma \vdash r_0 : B \qquad \Gamma, r : B \vdash_{\mathsf{c}} c : B' \,!\, \Delta' \qquad \Delta / \{\mathsf{op}_i\}_{1 \leqslant i \leqslant n} \subseteq \Delta'\end{array}}{\Gamma \vdash \mathbf{consolidate}_A\ \{\mathsf{op}_i((x,r) \mapsto c_i; (x,y) \mapsto k_i)\}_{1 \leqslant i \leqslant n}\ \mathbf{from}\ r_0\ \mathbf{with}\ c : A \,!\, \Delta \Rightarrow A \,!\, \Delta'}$$

- The handler is parametric in its return type A

Concurrent Handler: **consolidate**



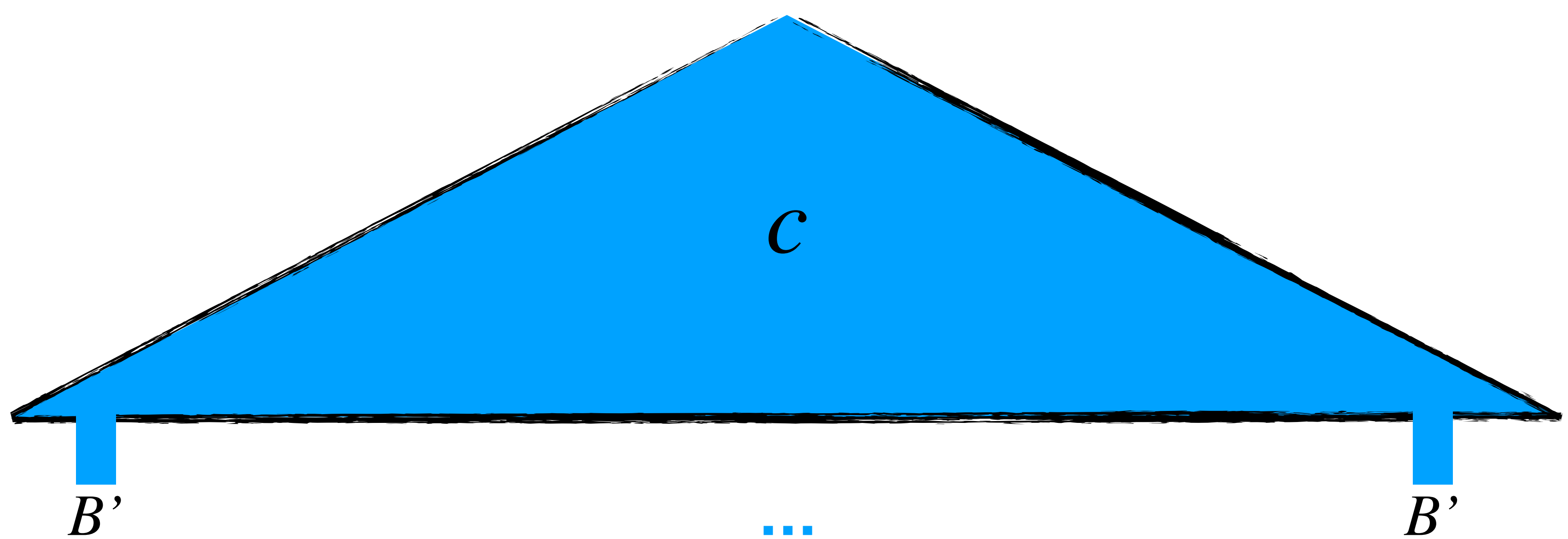
Concurrent Handler: **consolidate**



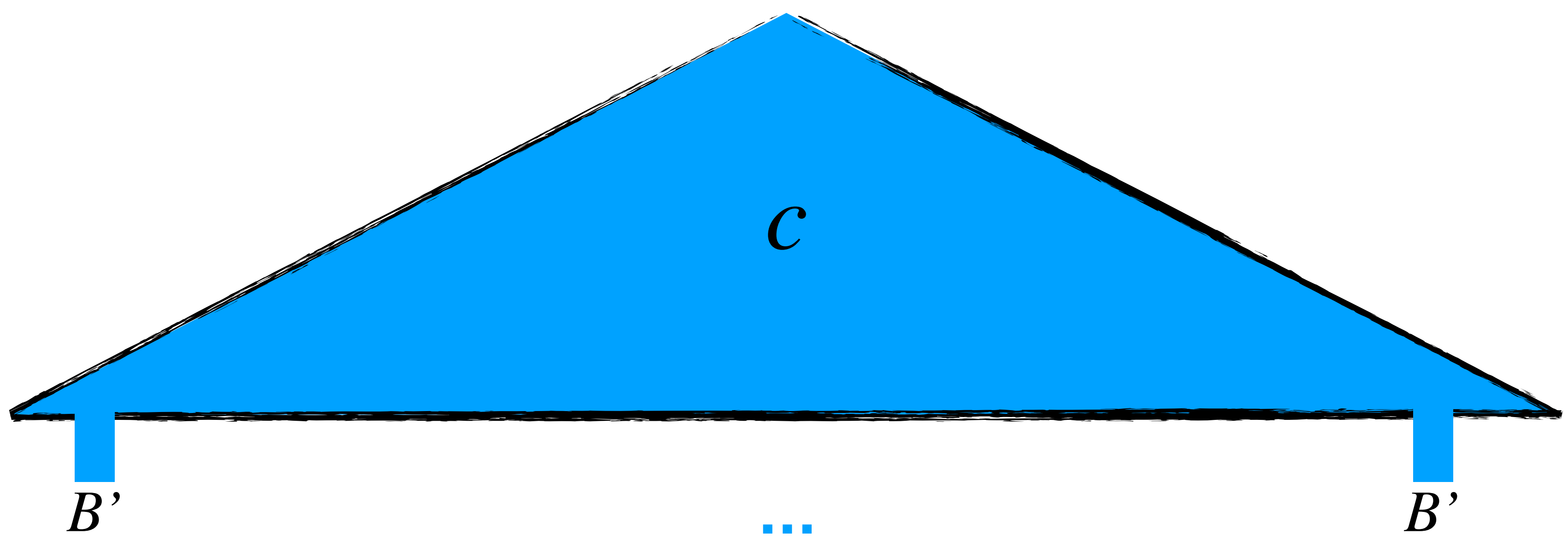
$$p_1 \oplus_1 (p_2 \oplus_2 (p_3 \oplus_1 r_0)) \mapsto r : B$$

Concurrent Handler: **consolidate**

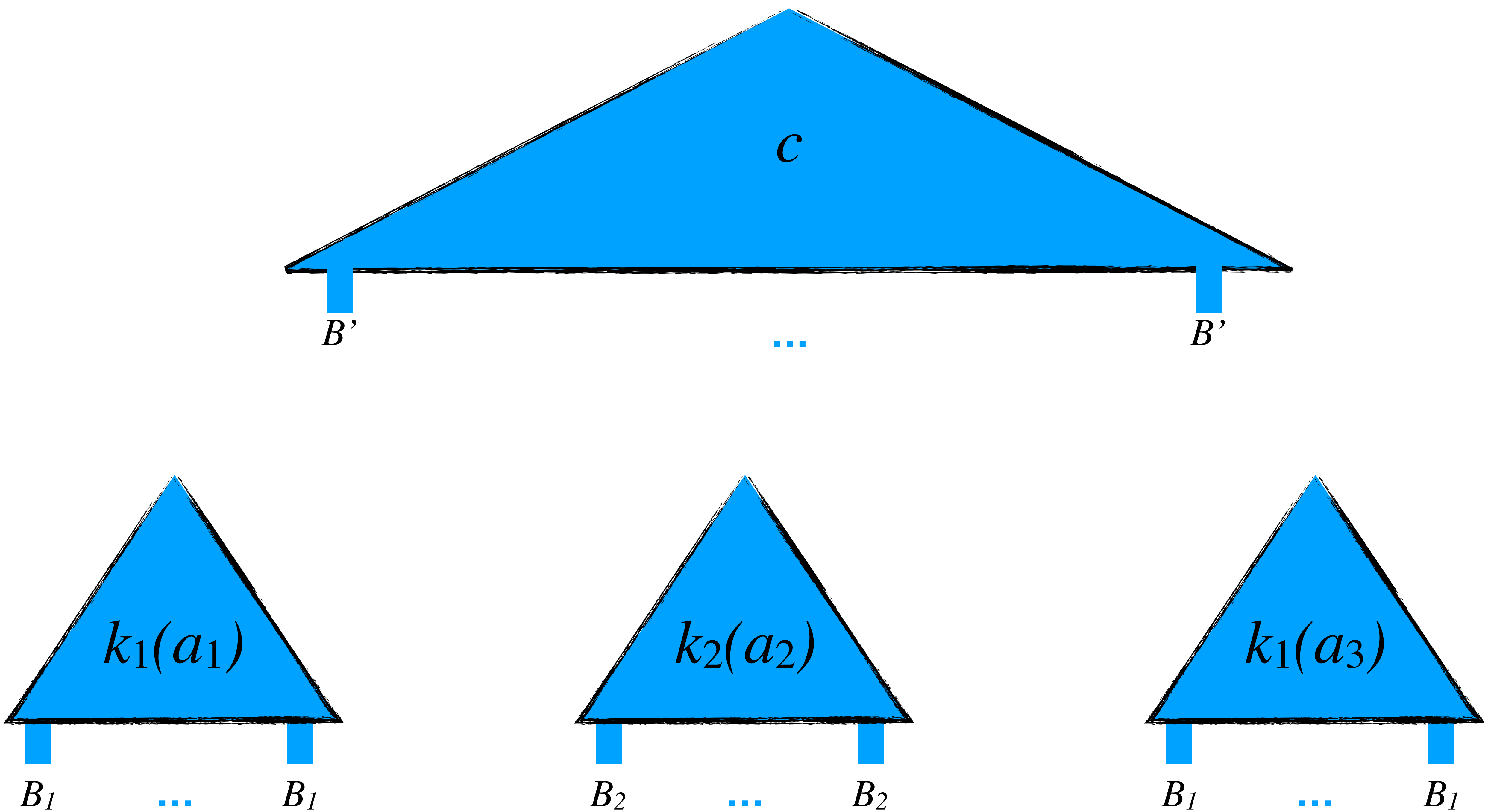
$$p_1 \oplus_1 (p_2 \oplus_2 (p_3 \oplus_1 r_0)) \mapsto r : B$$



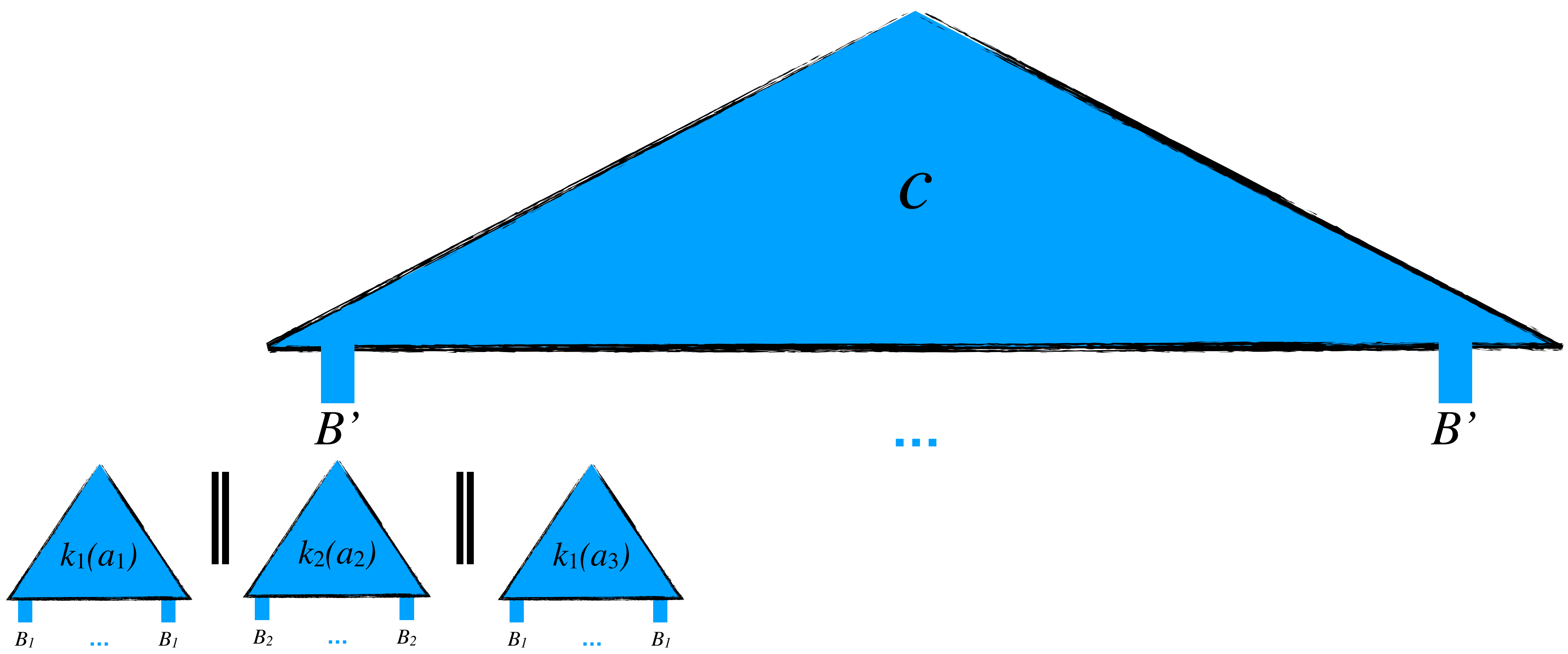
Concurrent Handler: **consolidate**



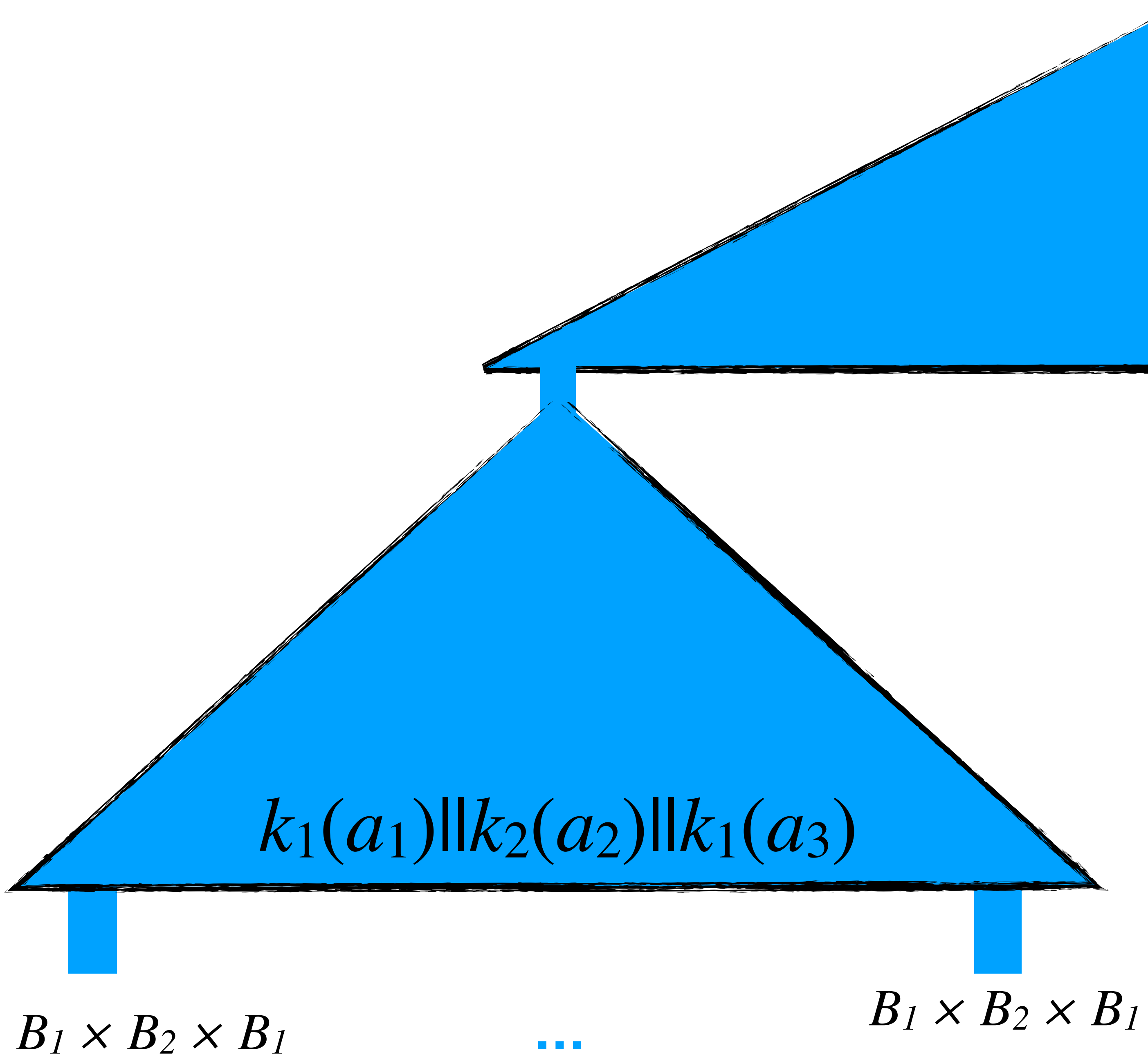
Concurrent Handler: **consolidate**



Concurrent Handler: **consolidate**



Concurrent Handler: **consolidate**



The Haskell Connection

The Haskell Connection

- Concurrent operations are related to the use of applicative programming for concurrency in Haskell (for example Haxl)

The Haskell Connection

- Concurrent operations are related to the use of applicative programming for concurrency in Haskell (for example Haxl)

- Applicative Functors are monoids wrt the Day convolution

$$(F \star G)\, a = \exists b, c\,.\, F(b) \times G(c) \times ((b \times c) \to a)$$

The Haskell Connection

- Concurrent operations are related to the use of applicative programming for concurrency in Haskell (for example Haxl)

- Applicative Functors are monoids wrt the Day convolution

$$(F \star G)\, a = \exists b\, .\, F(b \to a) \times G(b)$$

The Haskell Connection

- Concurrent operations are related to the use of applicative programming for concurrency in Haskell (for example Haxl)
- Applicative Functors are monoids wrt the Day convolution

$$(F \star G)\, a = \exists b \,.\, F(b \to a) \times G(b)$$

- $F_\Sigma{}^\star$ is closure of F_Σ wrt the Day convolution

The Haskell Connection

- Concurrent operations are related to the use of applicative programming for concurrency in Haskell (for example Haxl)

- Applicative Functors are monoids wrt the Day convolution

$$(F \star G)\, a = \exists b \,.\, F(b \to a) \times G(b)$$

- $F_\Sigma^\star$ is closure of F_Σ wrt the Day convolution

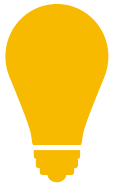
-

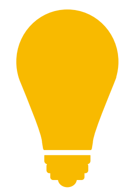
Free applicative functor over F_Σ	list of operations
$F_\Sigma^\star$	non-empy list of operations

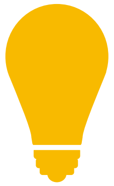
Effect of the Day

Concurrency

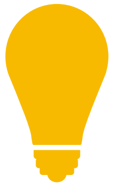
- Structured operations

- Structured operations 
- ChEff 

- Structured operations 
- ChEff 
- Choice of interleaving or native concurrency (or both)

- Structured operations 
- ChEff 
- Choice of interleaving or native concurrency (or both)

- Details of operational semantics 

- Structured operations 
- ChEff 
- Choice of interleaving or native concurrency (or both)

- Details of operational semantics 
- Different concurrent handlers.
 Need to unify.