Optimal Effects via Token-Passing

ICFP SRC 2025-10-14

Marvin Borner (Undergraduate)

University of Tübingen

ordered execution (effectful)
in parallel to
unordered reduction (pure)

ordered execution (effectful)
in parallel to
unordered reduction (pure)
via token-passing

Optimality

= Call-by-Need + Redex Sharing = Maximal Laziness

Optimality

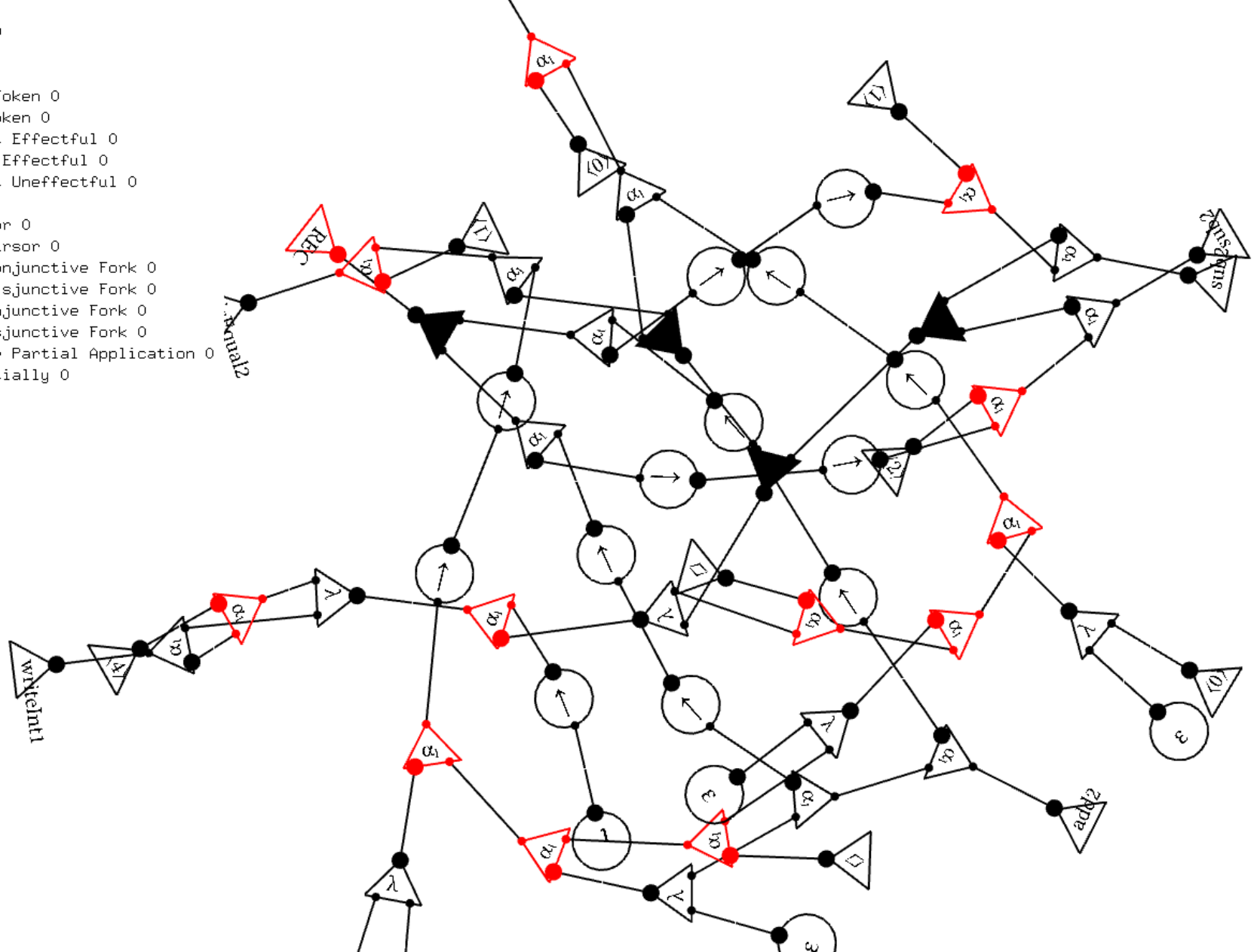
= Call-by-Need + Redex Sharing = Maximal Laziness

```
let f = x => x + factorial(100)
in  f(0) + f(1)
```

Optimality from Graph Encoding: Interaction Nets

Optimality from Graph Encoding: *Effectful* Interaction Nets

Duplicate 0
Annihilate 0
Erase 0
Token 0
  Redirect Token 0
  Reflect Token 0
  Infer Left Effectful 0
  Infer Top Effectful 0
  Infer Left Uneffectful 0
Effectful 0
  Apply Actor 0
  Apply Recursor 0
  Execute Conjunctive Fork 0
  Execute Disjunctive Fork 0
  Return Conjunctive Fork 0
  Return Disjunctive Fork 0
  Initialize Partial Application 0
  Apply Partially 0



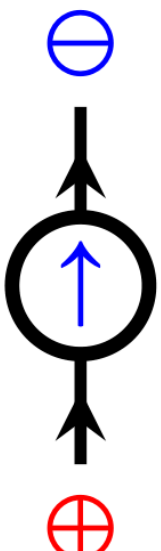
Interaction Nets are Inherently Unordered

Main Idea

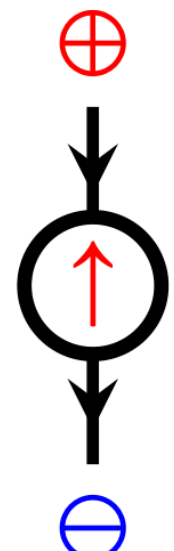
Tokens Traverse the Net and Execute Actors

Main Idea

Tokens Traverse the Net and Execute Actors

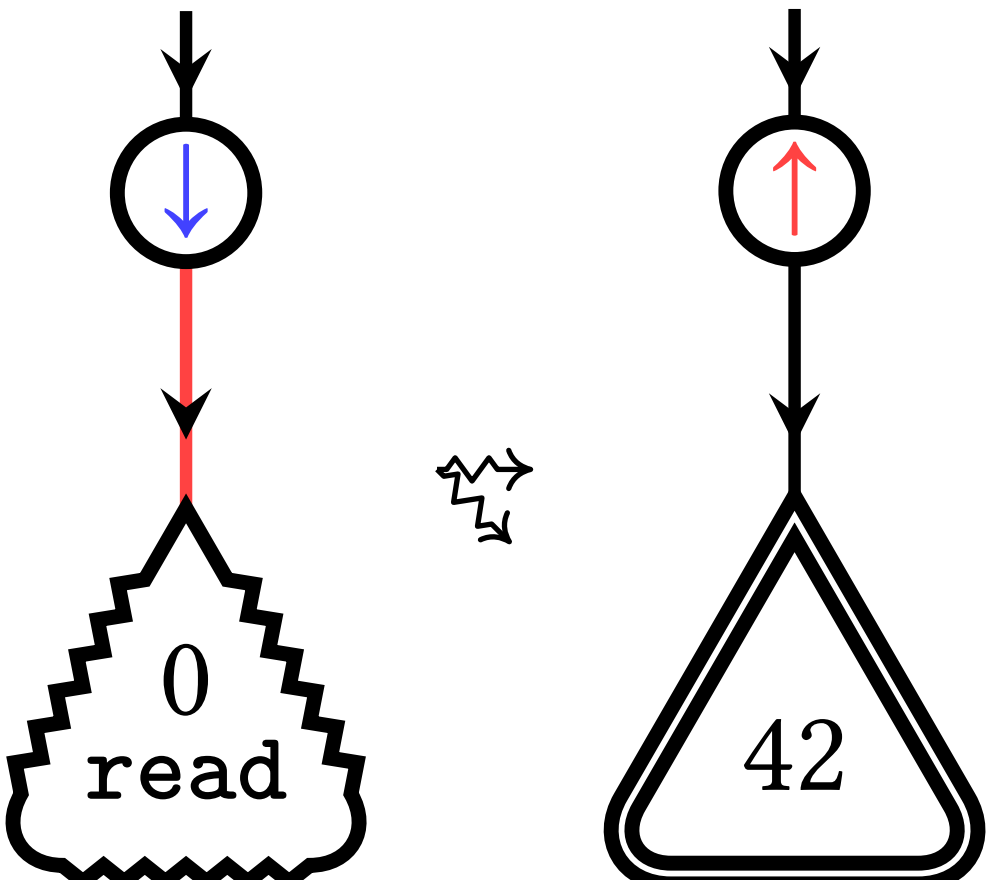


Token

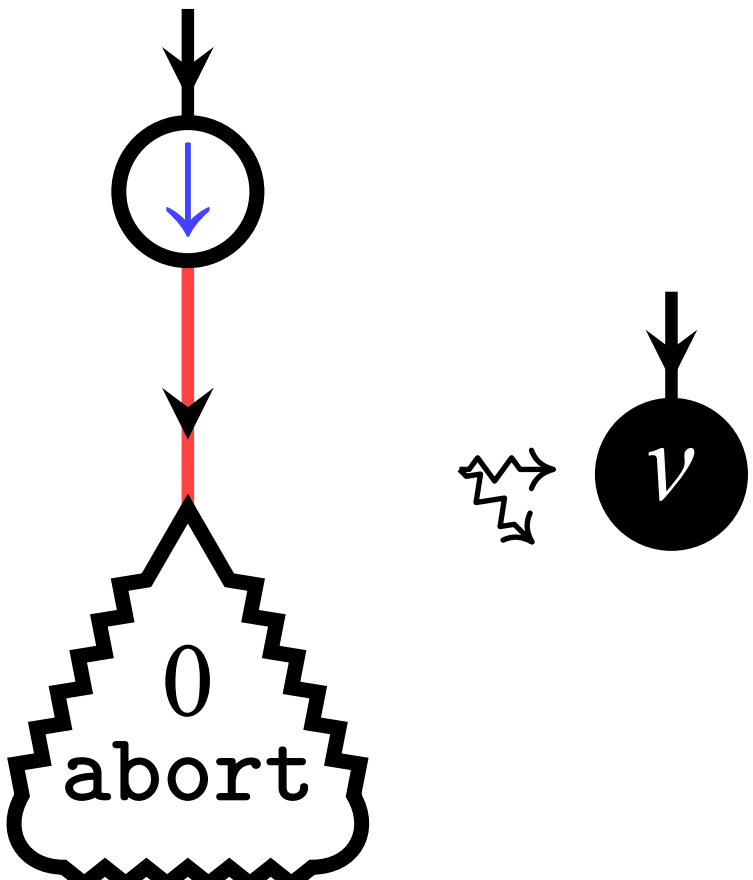


Cotoken

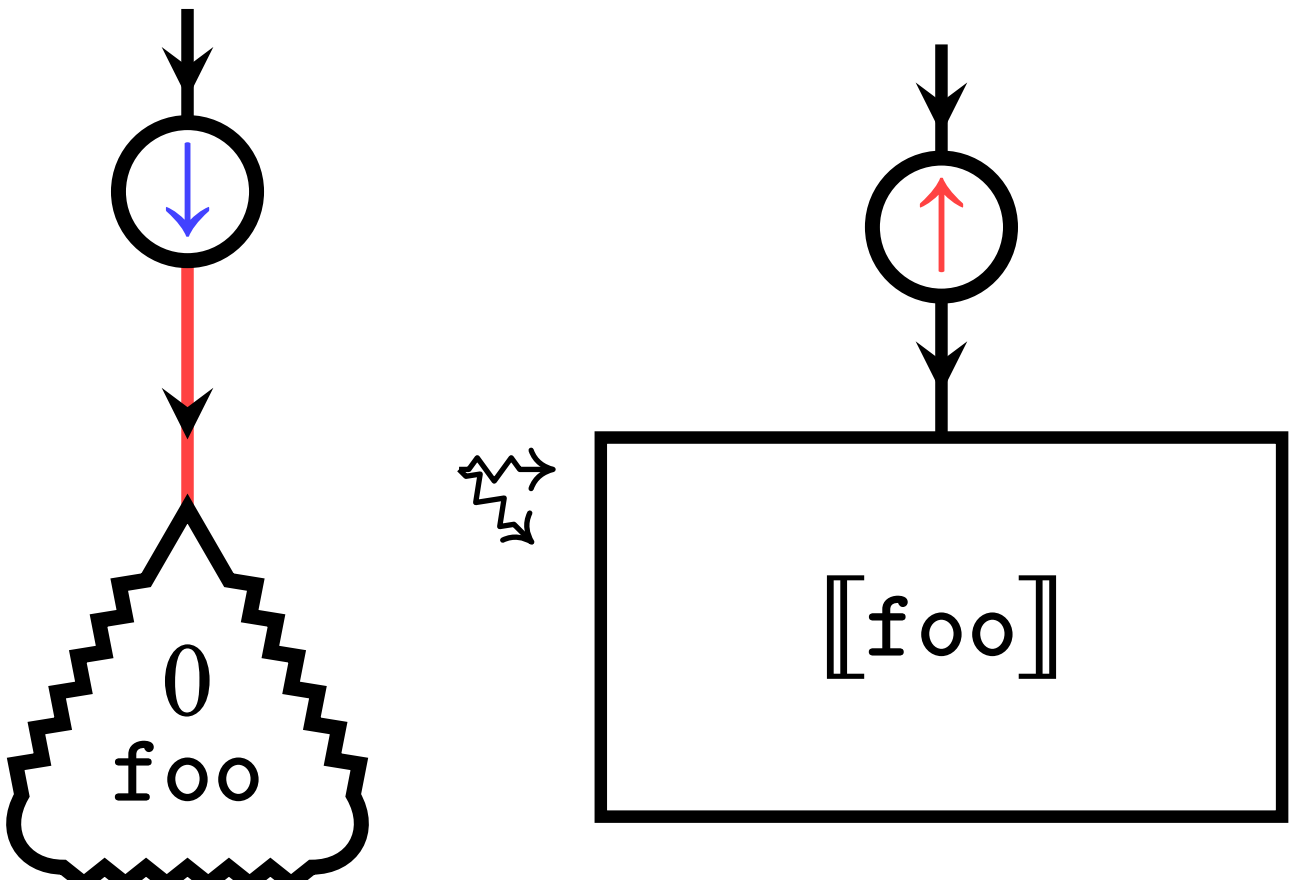
Execution of Actors: Read



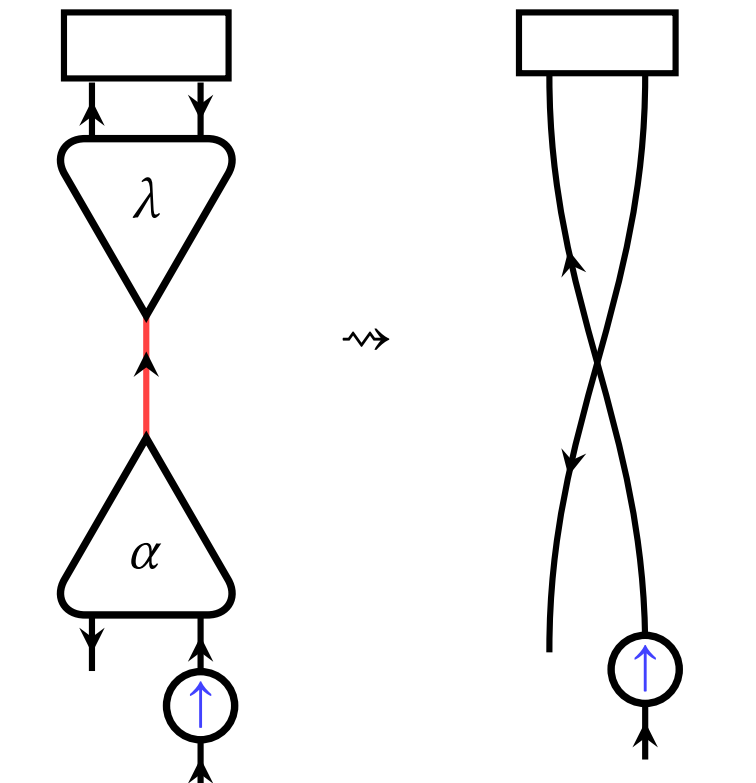
Execution of Actors: Abort Thread



Execution of Actors: Recursion/FFI/RPC/...



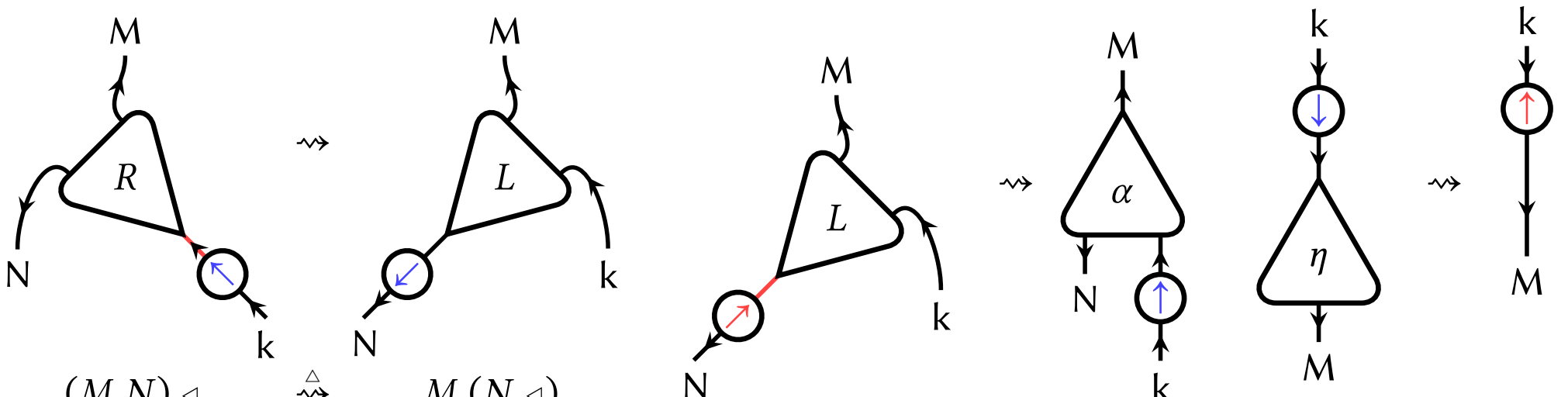
Reduction Paves the Path for Execution



Graph Rewrite Rules Redirect the Token

Graph Rewrite Rules Redirect the Token

Monadic



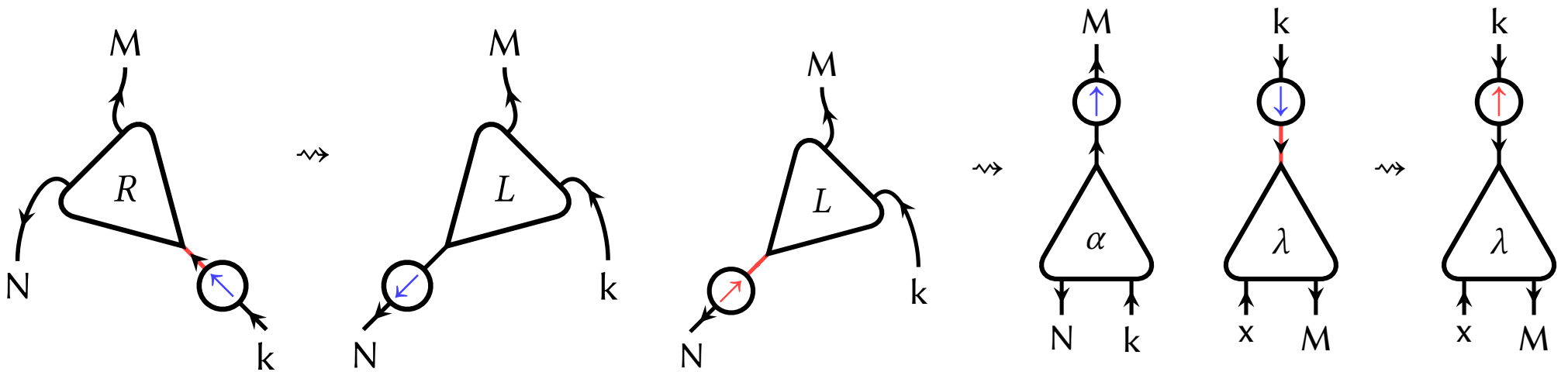
$(\underline{M}\ \underline{N})\ \triangleleft \overset{\triangle}{\rightsquigarrow} \underline{M}\ (\underline{N}\ \triangleleft)$

$(N \gg\!= M)\ \triangleleft \overset{\triangle}{\rightsquigarrow} (N\ \triangleleft) \gg\!= M$

$((N\ \triangleright) \gg\!= M) \overset{\triangle}{\rightsquigarrow} (M\ N)\ \triangleleft \quad (\texttt{unit}\ M)\ \triangleleft \overset{\triangle}{\rightsquigarrow} M\ \triangleright$

Graph Rewrite Rules Redirect the Token

Direct Call-by-Value



$(\underline{M}\ \underline{N})\ \triangleleft \overset{\triangle}{\rightsquigarrow} \underline{M}\ (\underline{N}\ \triangleleft)$

$(N \gg\!= M)\ \triangleleft \overset{\triangle}{\rightsquigarrow} (N\ \triangleleft) \gg\!= M$

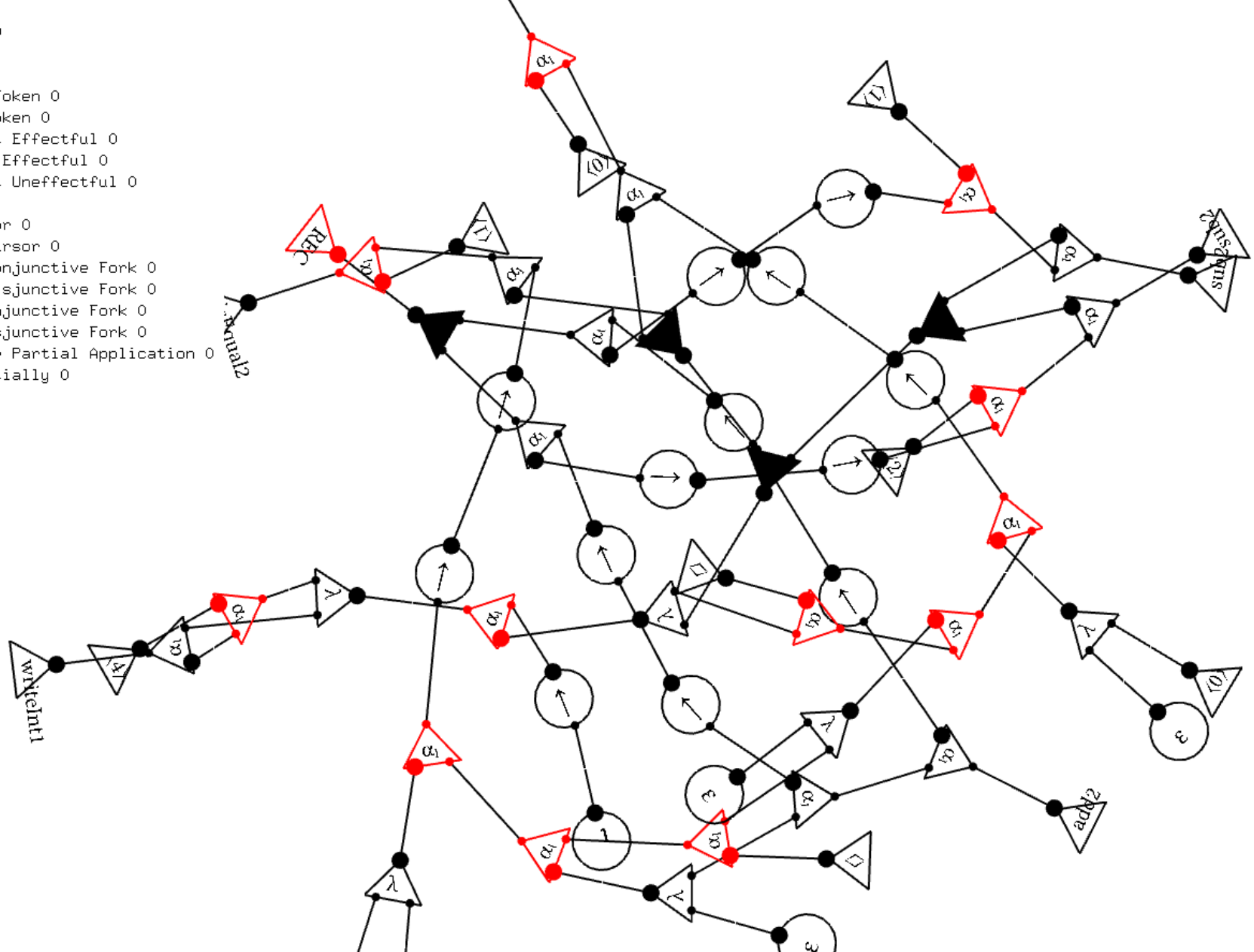
$\underline{M}\ (N\ \triangleright) \overset{\triangle}{\rightsquigarrow} (\underline{M}\ \triangleleft)\ N$

$(M\ \triangleright)\ N \overset{\triangle}{\rightsquigarrow} (M\ N)\ \triangleleft$

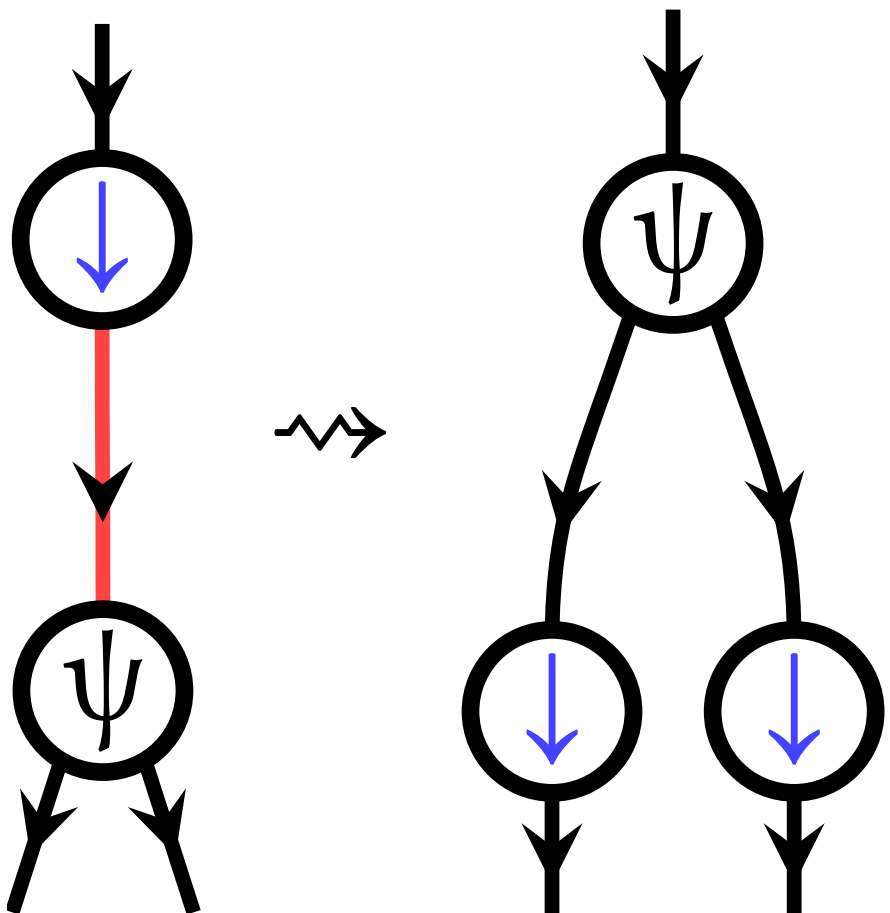
$(\underline{\lambda x.M})\ \triangleleft \overset{\triangle}{\rightsquigarrow} (\lambda x.\underline{M})\ \triangleright$

All While Retaining the One-Step Diamond Property (Strong Confluence)

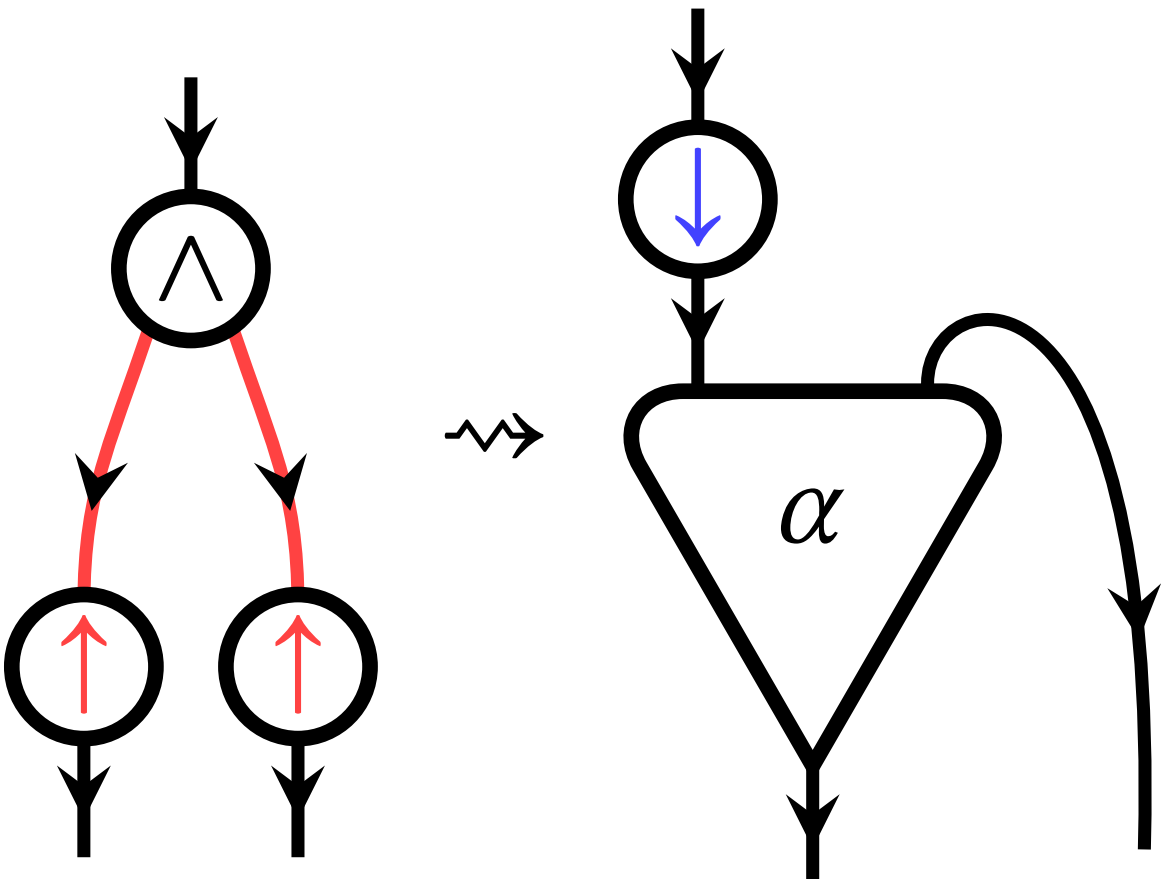
Duplicate 0
Annihilate 0
Erase 0
Token 0
  Redirect Token 0
  Reflect Token 0
  Infer Left Effectful 0
  Infer Top Effectful 0
  Infer Left Uneffectful 0
Effectful 0
  Apply Actor 0
  Apply Recursor 0
  Execute Conjunctive Fork 0
  Execute Disjunctive Fork 0
  Return Conjunctive Fork 0
  Return Disjunctive Fork 0
  Initialize Partial Application 0
  Apply Partially 0



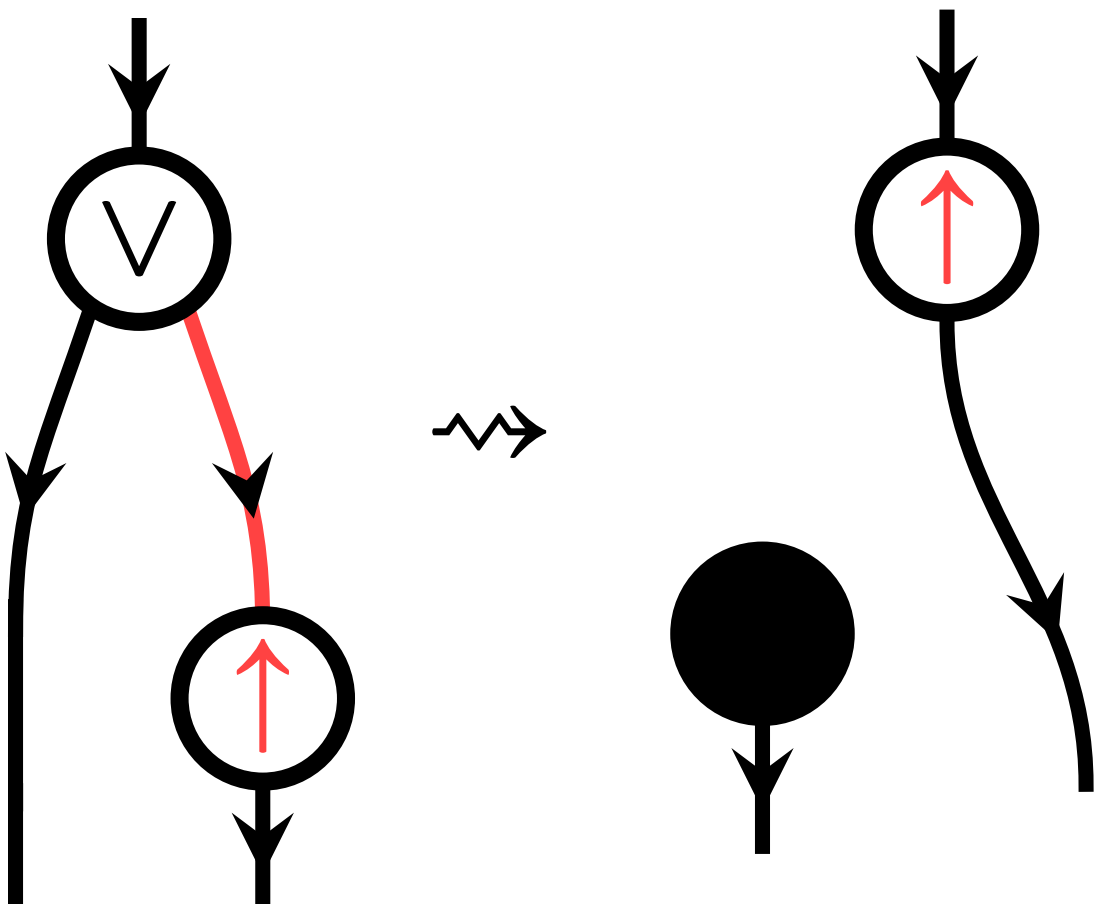
Multiple Tokens ⇔ Multiple Threads: `fork`



Multiple Tokens $\Leftrightarrow$ Multiple Threads: `join`



Multiple Tokens ⇔ Multiple Threads: `race`



Summary

- **Ordered token traversal happens in parallel to unordered graph reduction**
- Optimality & strong confluence (re. parallelism) remain

More (skipped)

- **Proofs** (monad laws, strong confluence, …)
- Polarity **type system** (token <> cotoken)
- **Asynchronous** actions & their sharing requirements
- **Partial application** of actors
- Source language, **core calculus** & implementation
- **Bookkeeping** via actors (defunctionalization)
- **Results**: Efficiency, available parallelism

Thank You!