Faucet: a user-level, modular technique for flow control in dataflow engines

Andrea Lattuada
Systems Group,
ETH Zürich

Frank McSherry
Unaffiliated

Zaheer Chothia
Systems Group,
ETH Zürich

Problem

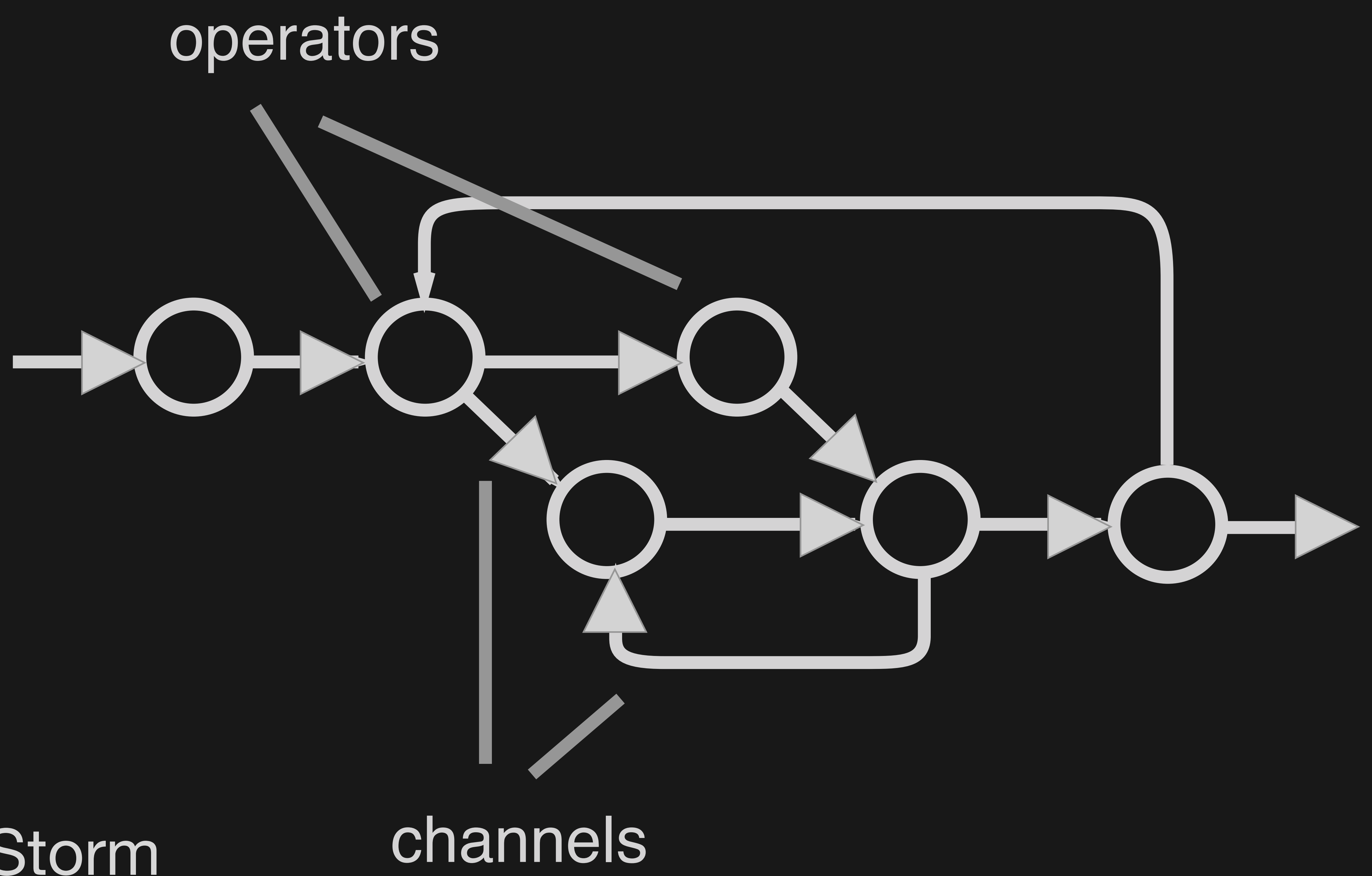
RAM exhaustion due to buffered *intermediate results*

Our Solution

- no system-level general strategy
- *application-driven* scheduling

10-100x memory savings for 15-25% runtime overhead

Dataflow model



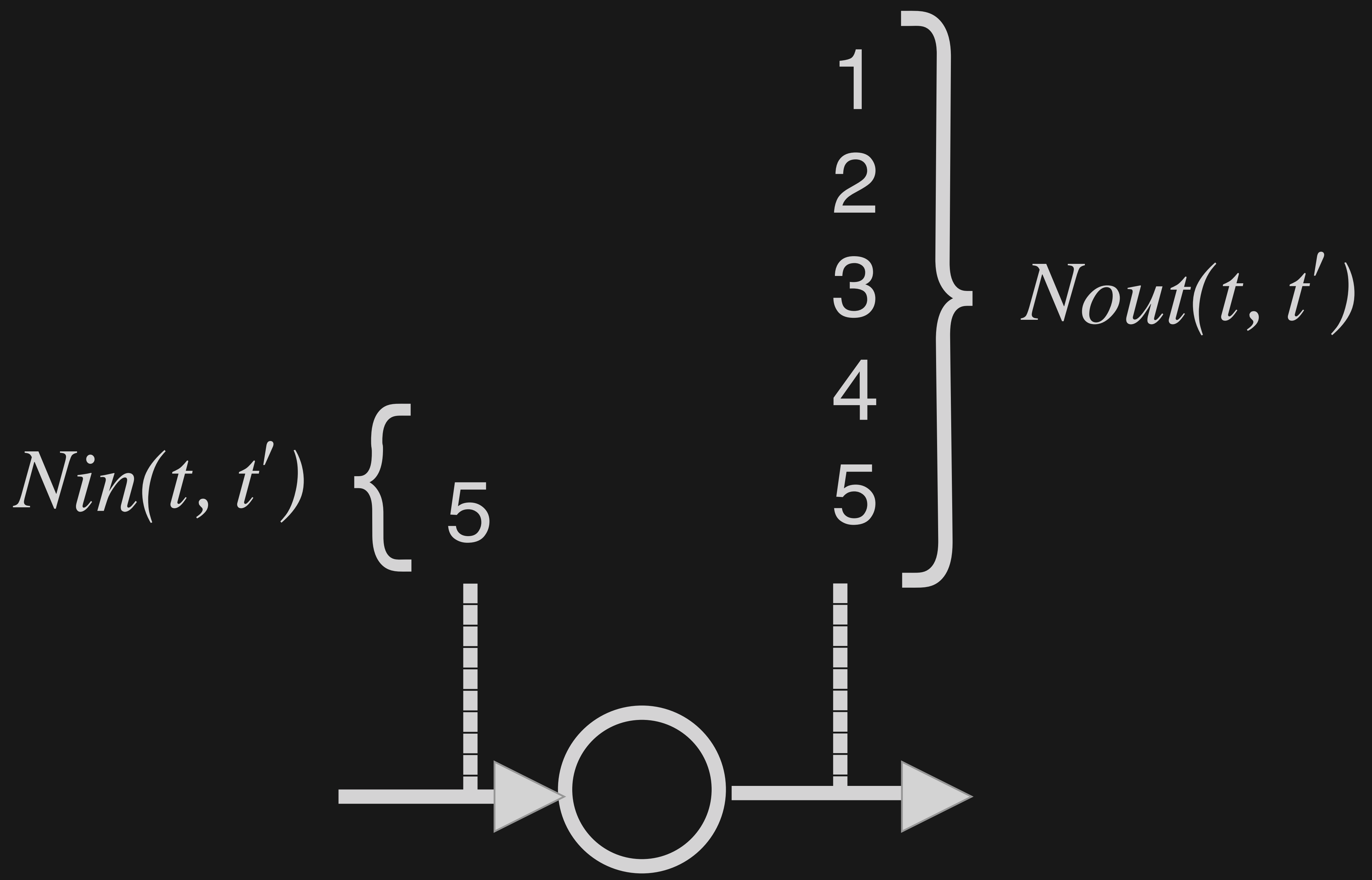
Storm

Flink

Naiad

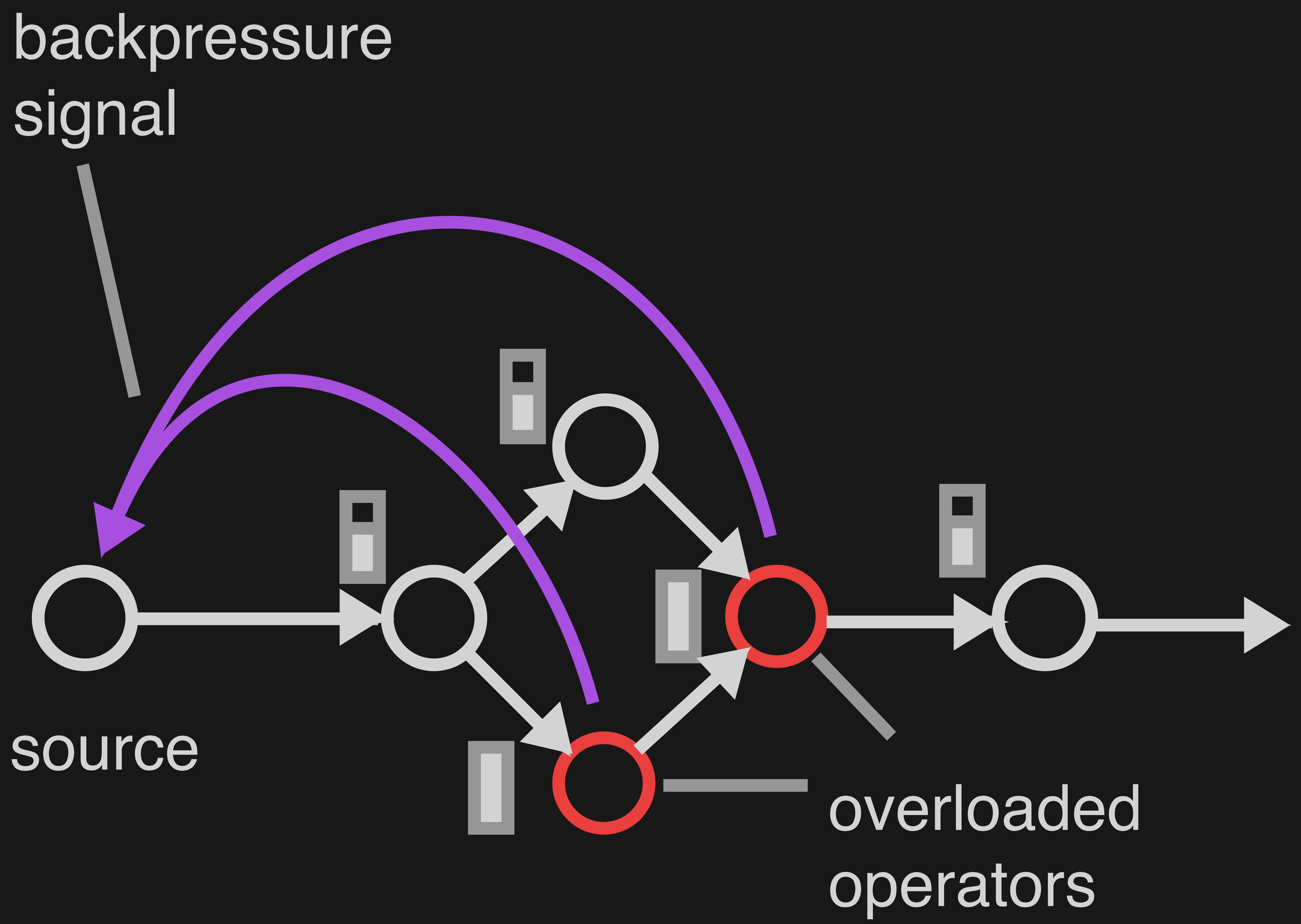
Source of the problem

Rate imbalance



```
flat_map(|x| [1, …, x])
```

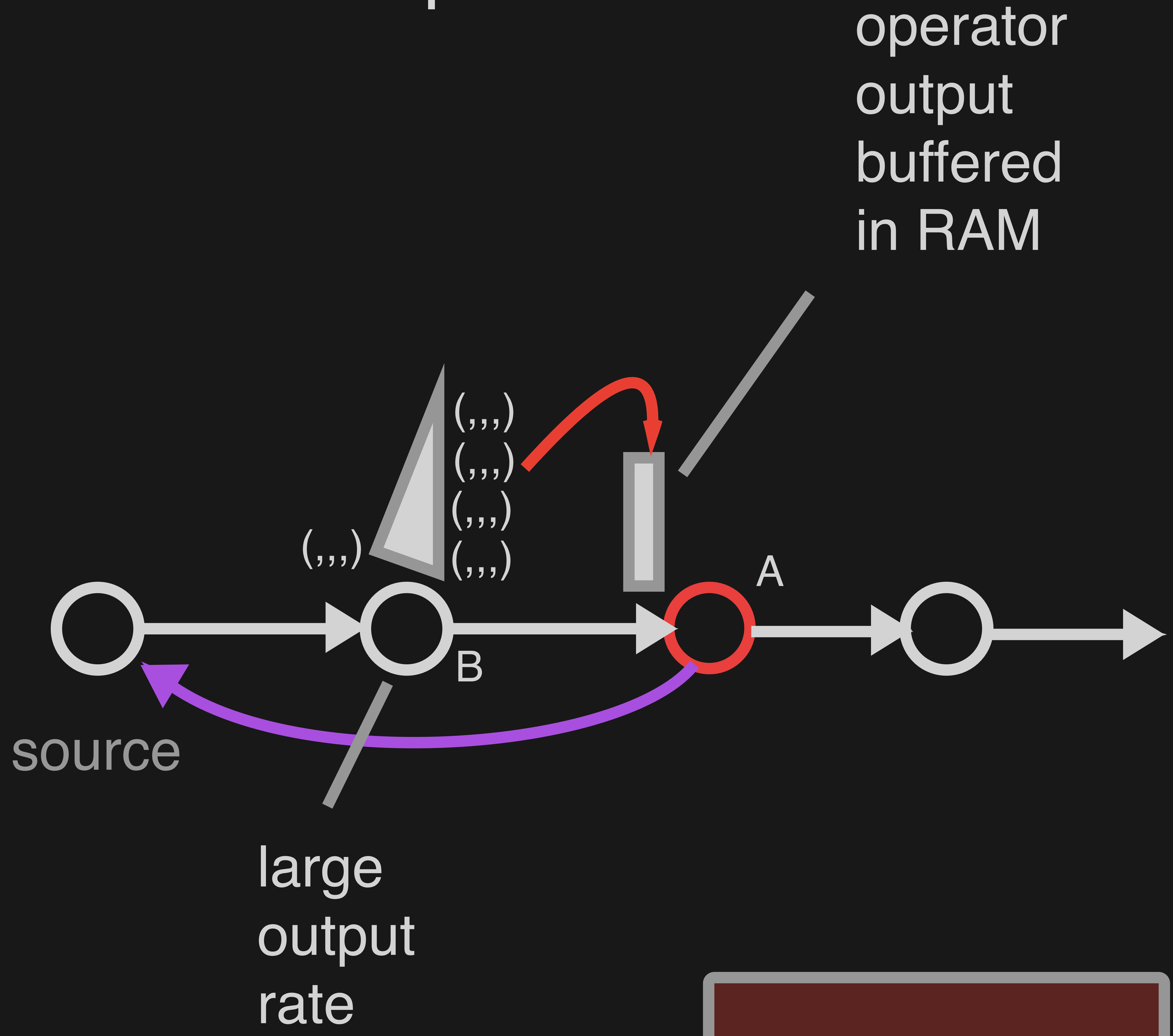
Existing approach #1 - Source backpressure



Storm

Heron

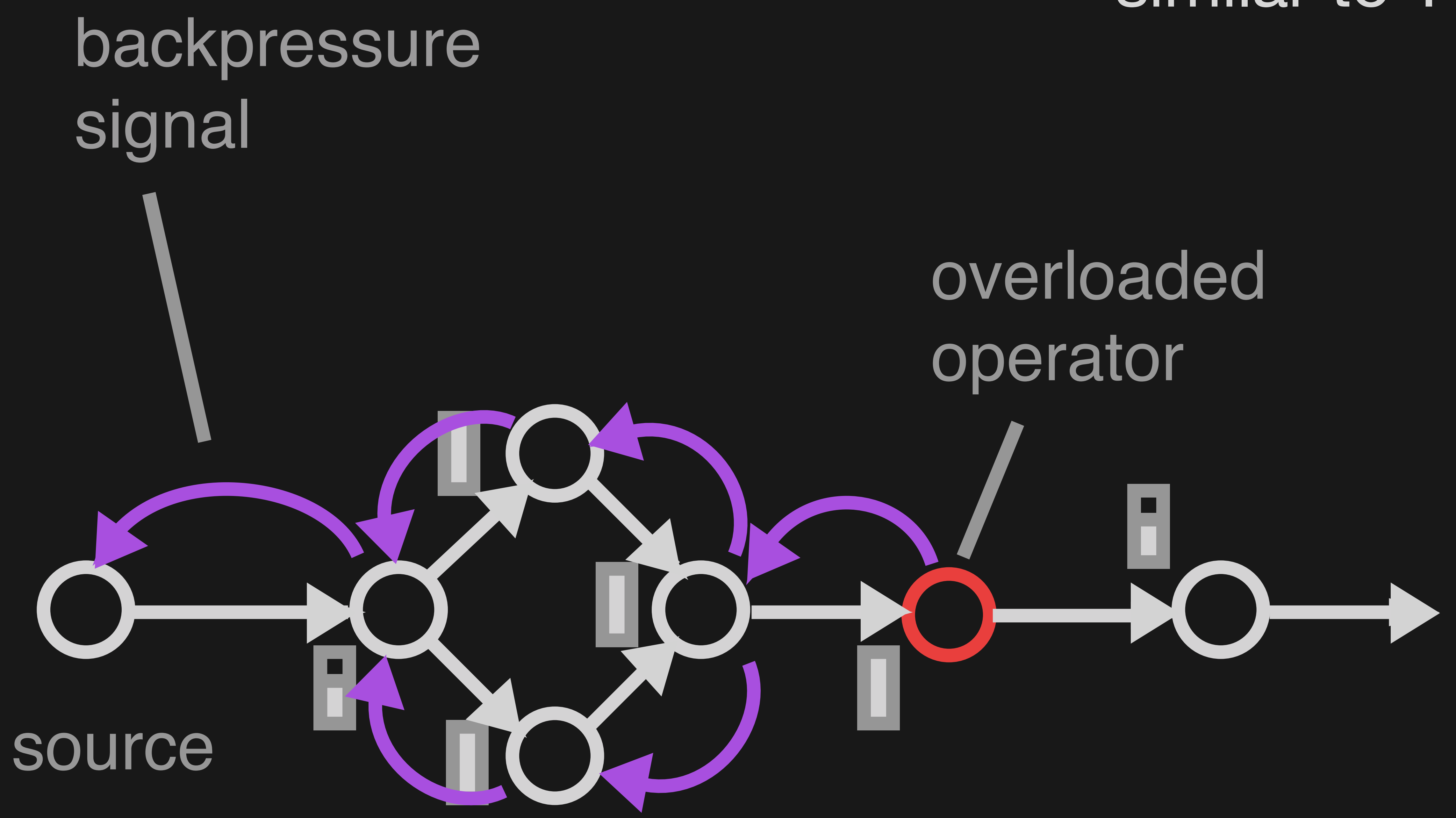
Spark streaming

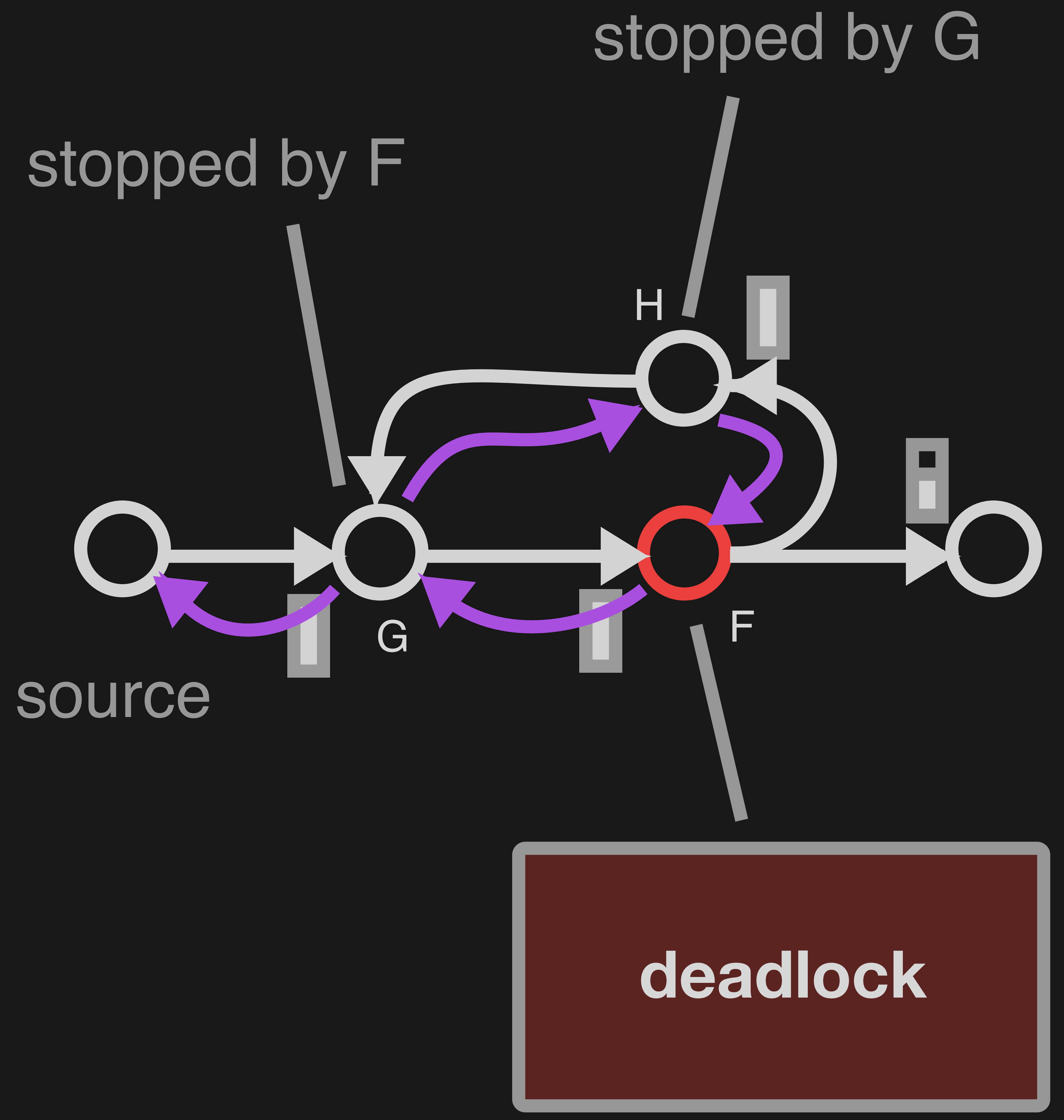


RAM
exhaustion

Existing approach #2 - Edge-by-edge backpressure

similar to TCP flow control





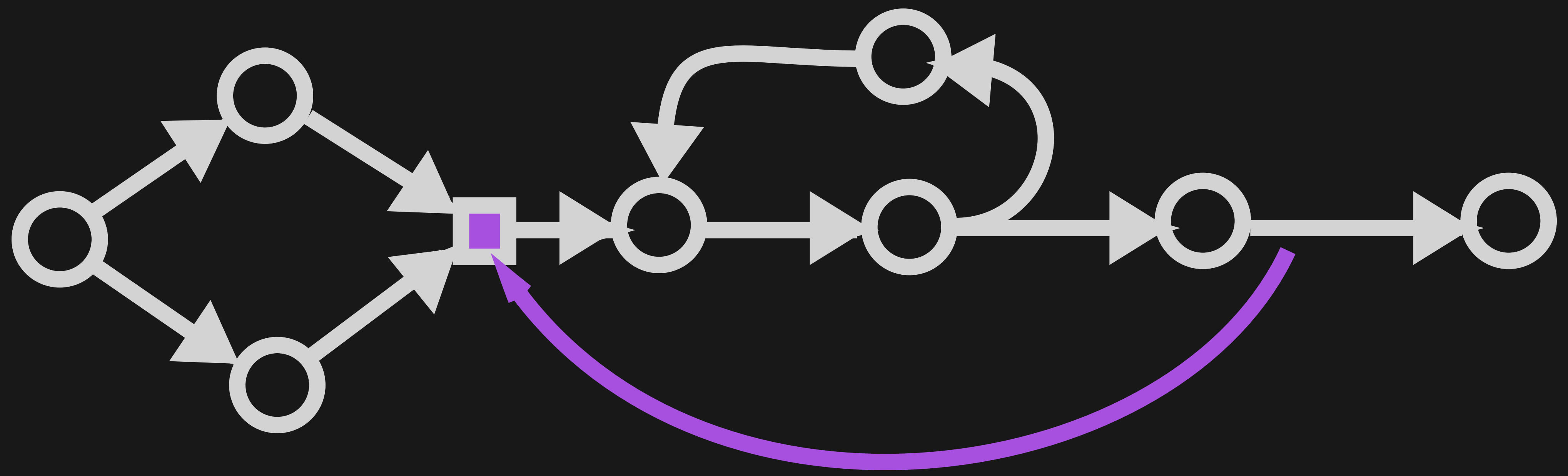
Akka Streams

Flink

Our approach - Faucet

based on *Timely Dataflow's concepts*

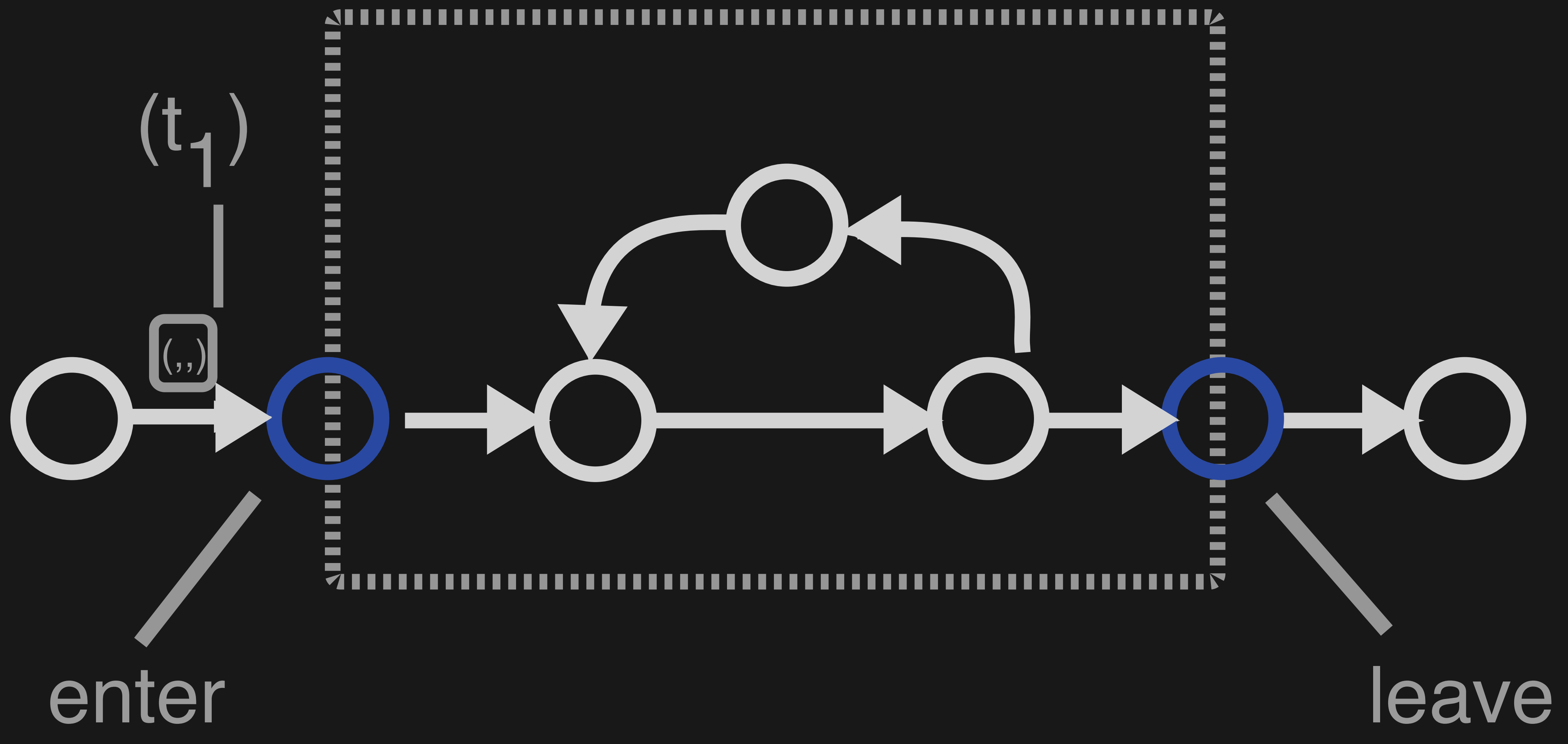
- no fine-grained signal
- *track completion* of a batch of tuples



control scheduling to *limit intermediate results*

Foundation - Timely Dataflow's Progress Tracking

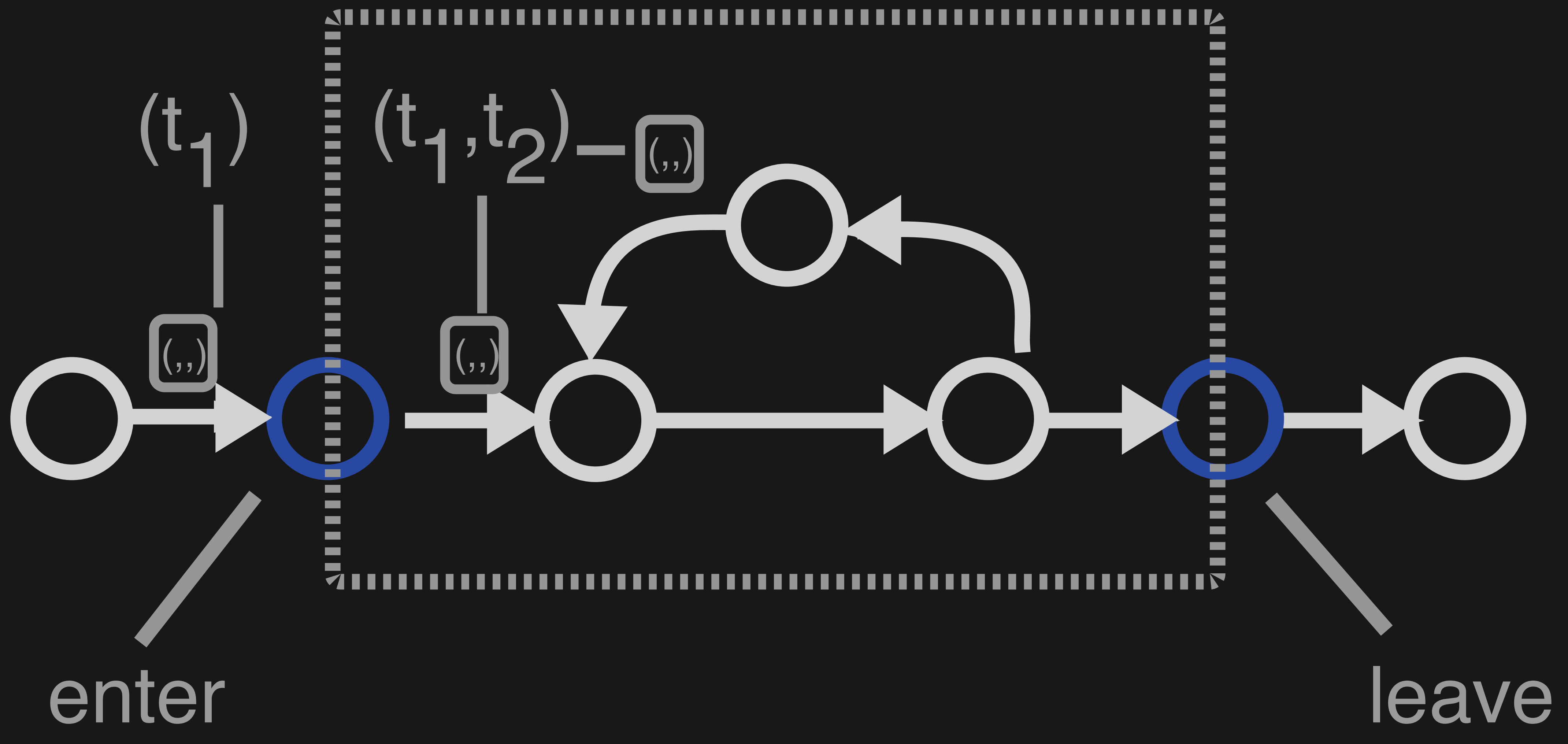
Scopes	nested operator structure
Timestamps	tuple metadata



Foundation - Timely Dataflow’s Progress Tracking

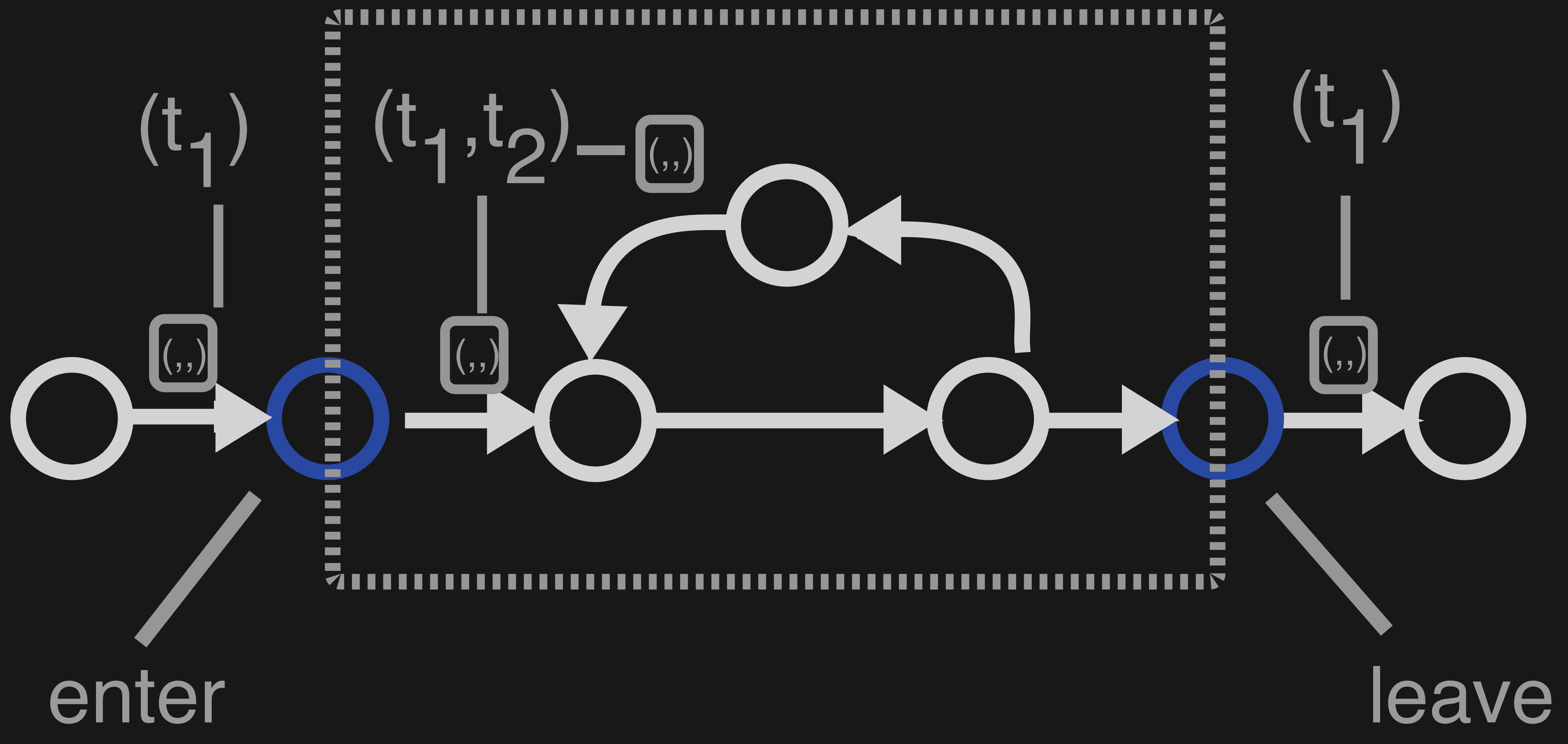
Scopes nested operator structure

Timestamps tuple metadata



Foundation - Timely Dataflow's Progress Tracking

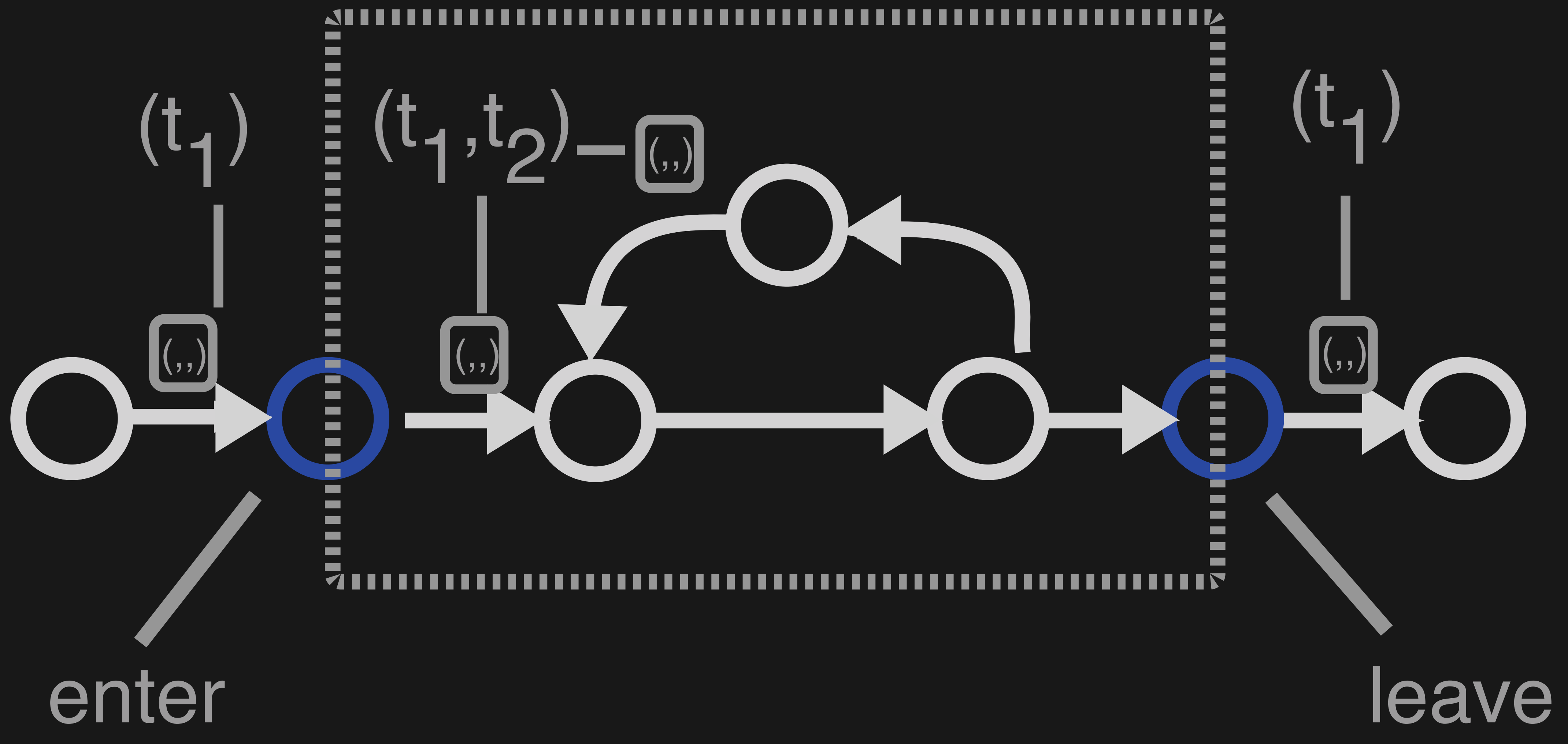
Scopes	nested operator structure
Timestamps	tuple metadata



Foundation - Timely Dataflow's Progress Tracking

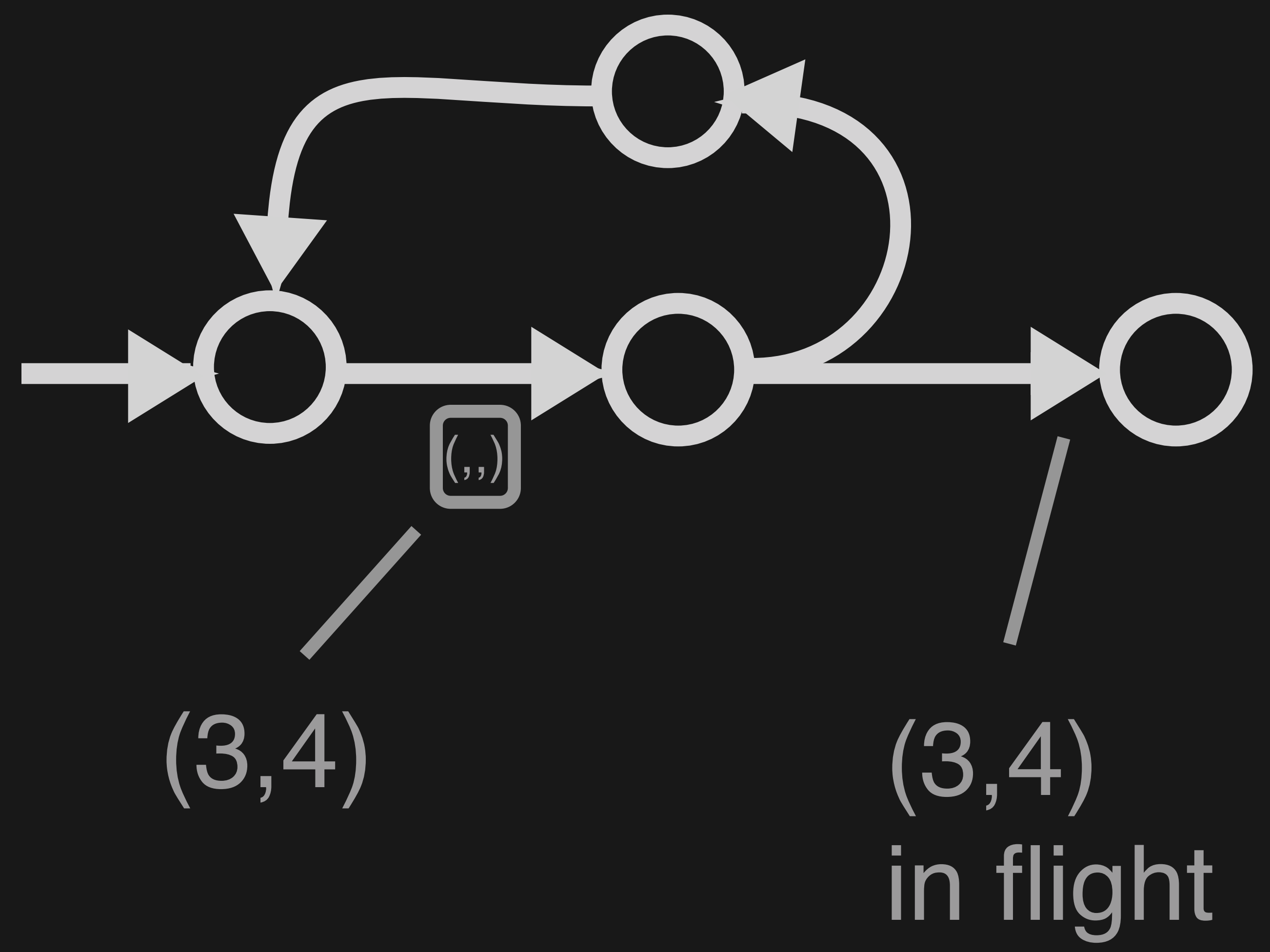
Scopes nested operator structure

Timestamps tuple metadata

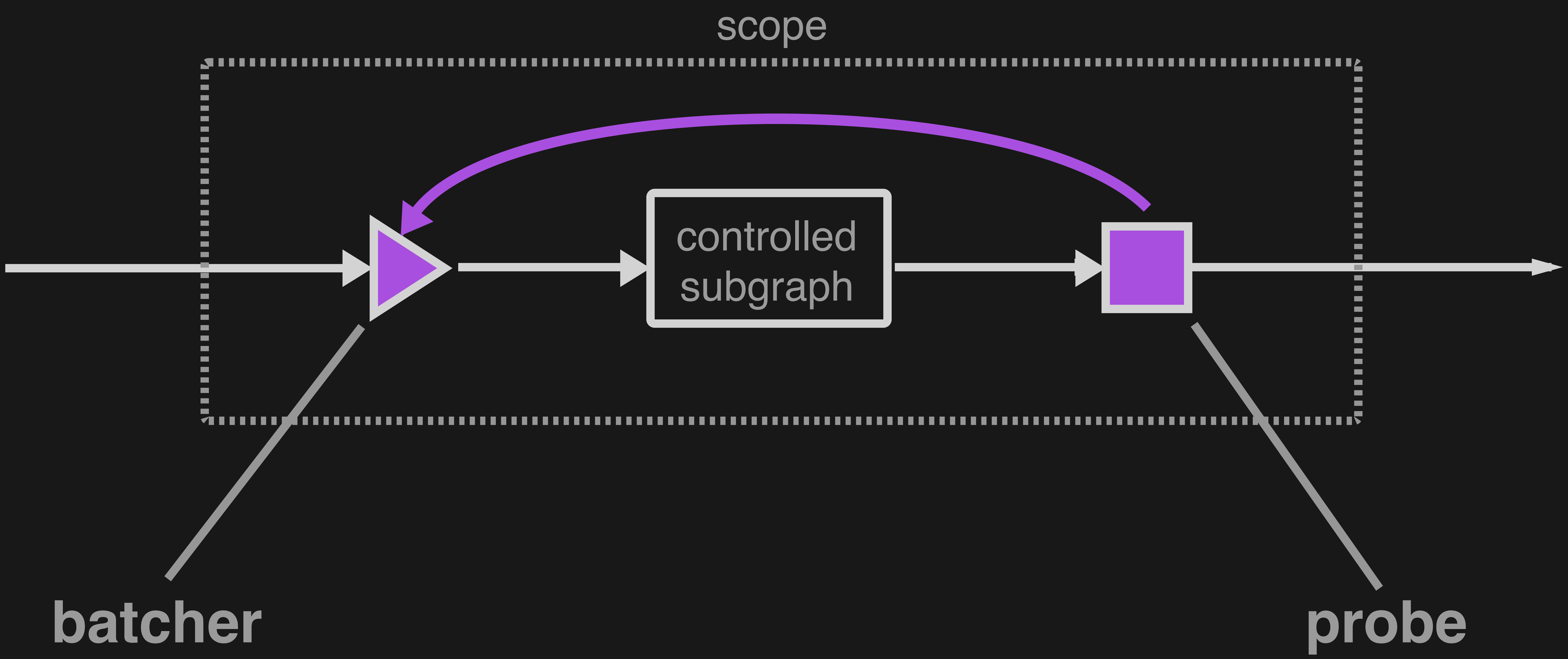


Progress Tracking

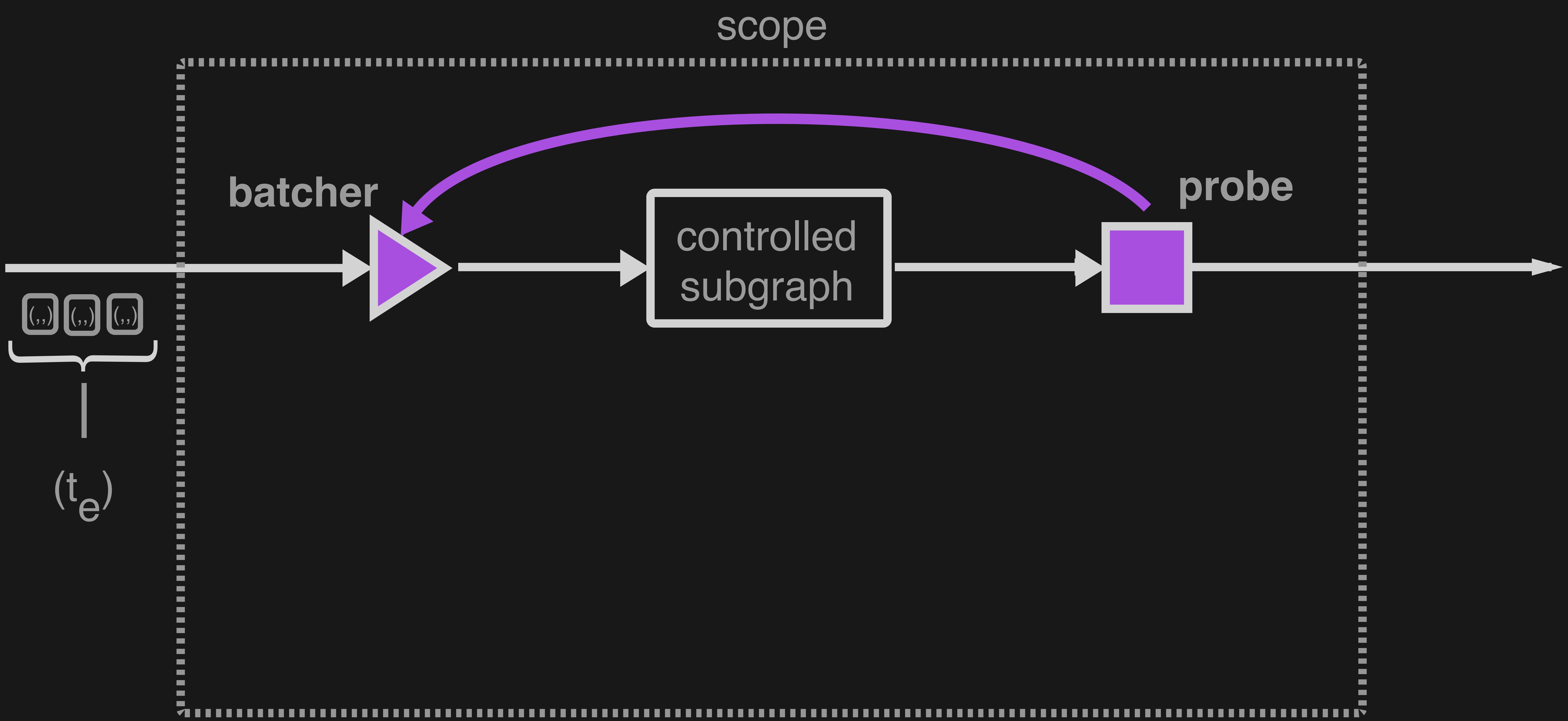
tracks pending timestamps



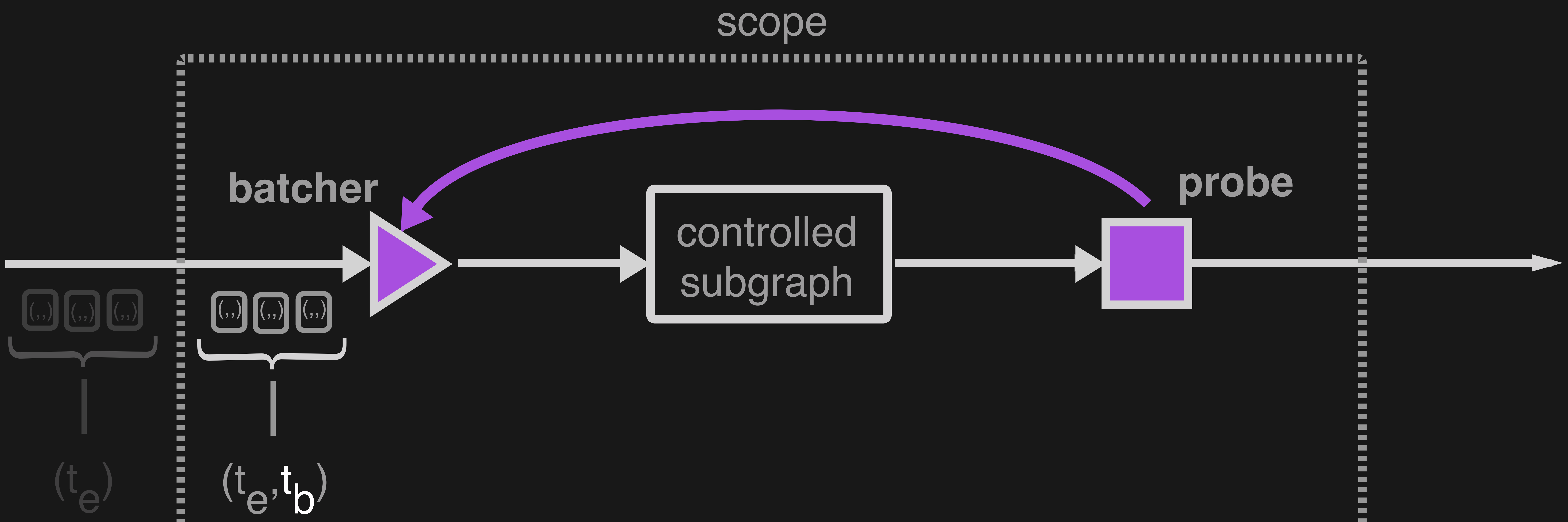
Faucet - Track batches of *intermediate results*



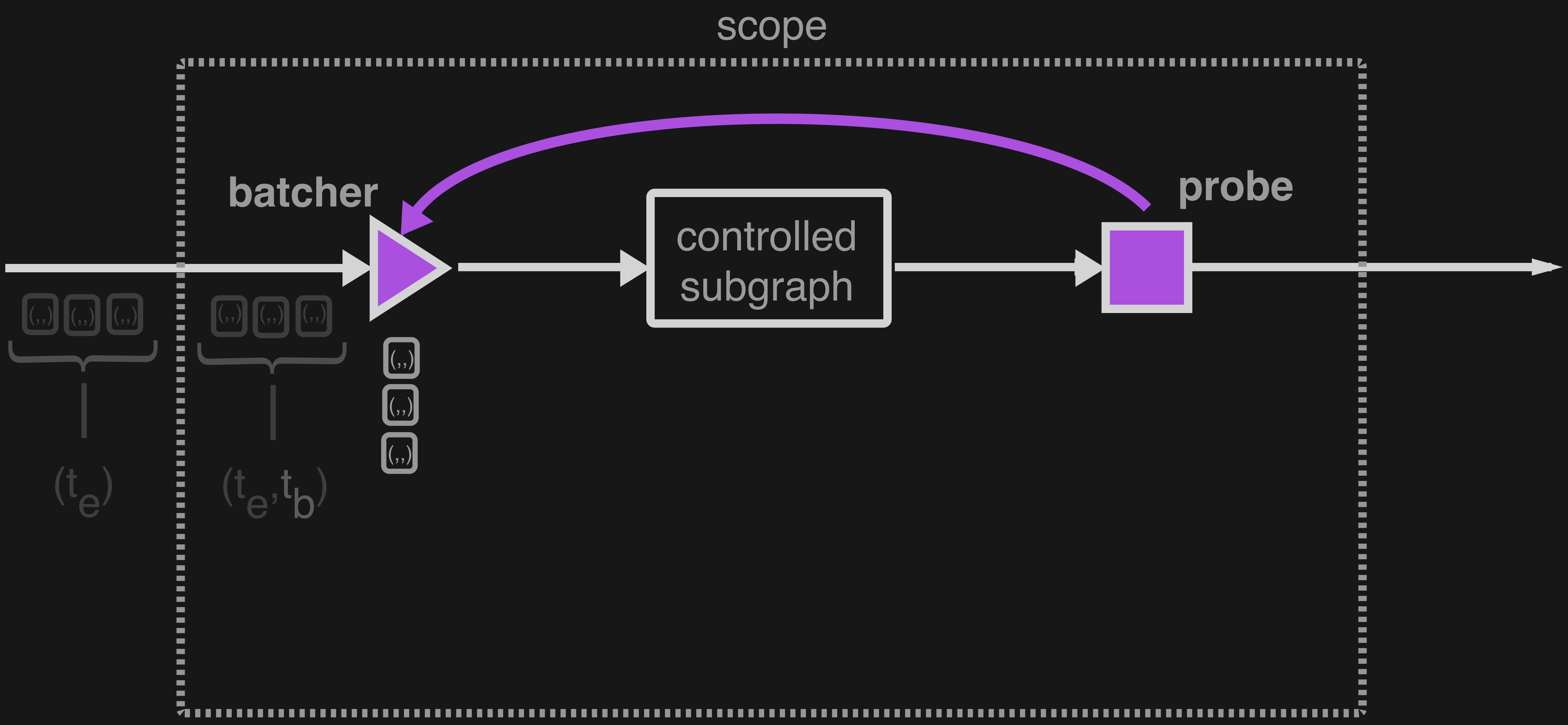
Faucet - Track batches of *intermediate results*



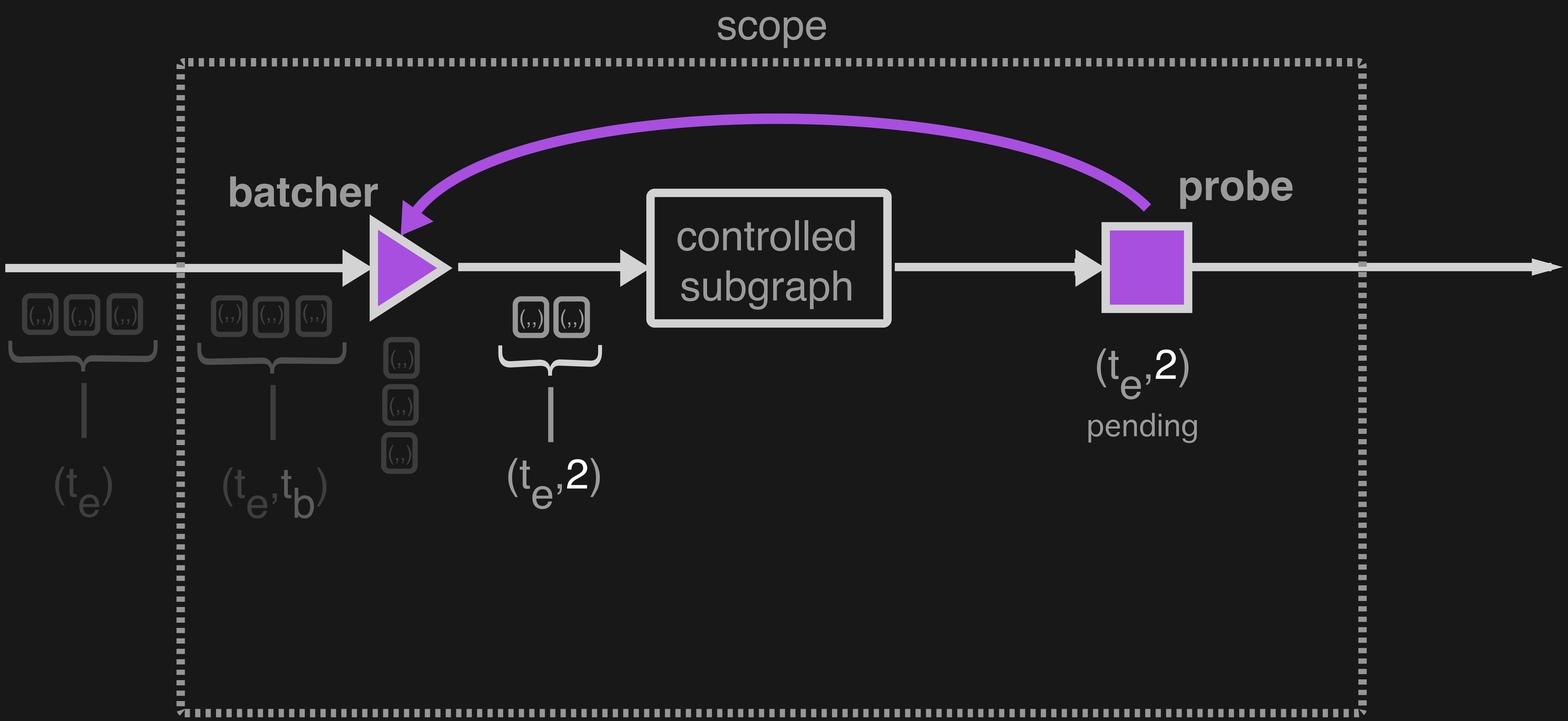
Faucet - Track batches of *intermediate results*



Faucet - Track batches of *intermediate results*

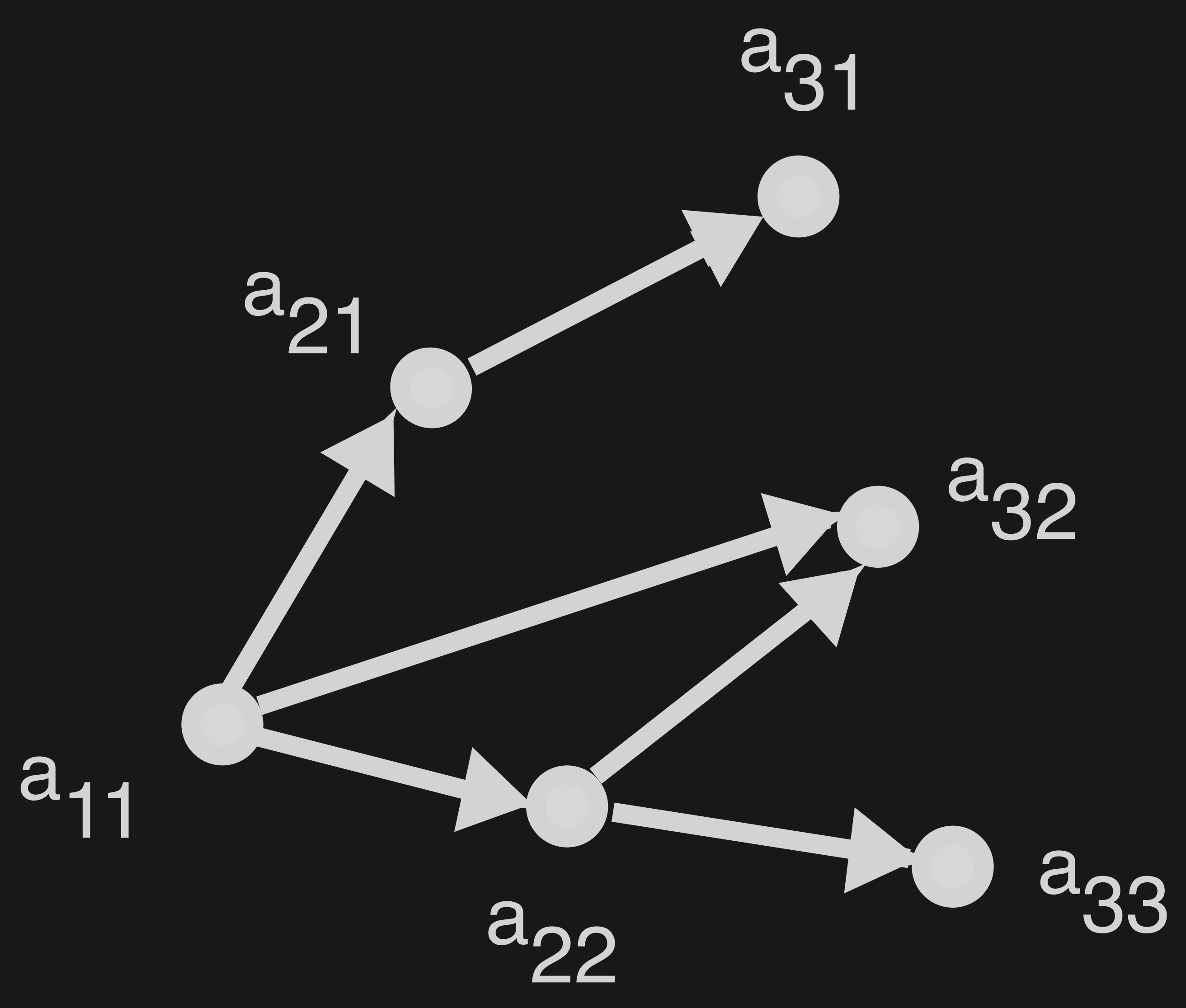


Faucet - Track batches of *intermediate results*



Example - Enumerate triangles in a directed graph

H. Q. Ngo, C. Ré, and A. Rudra - Generic Join



input graph

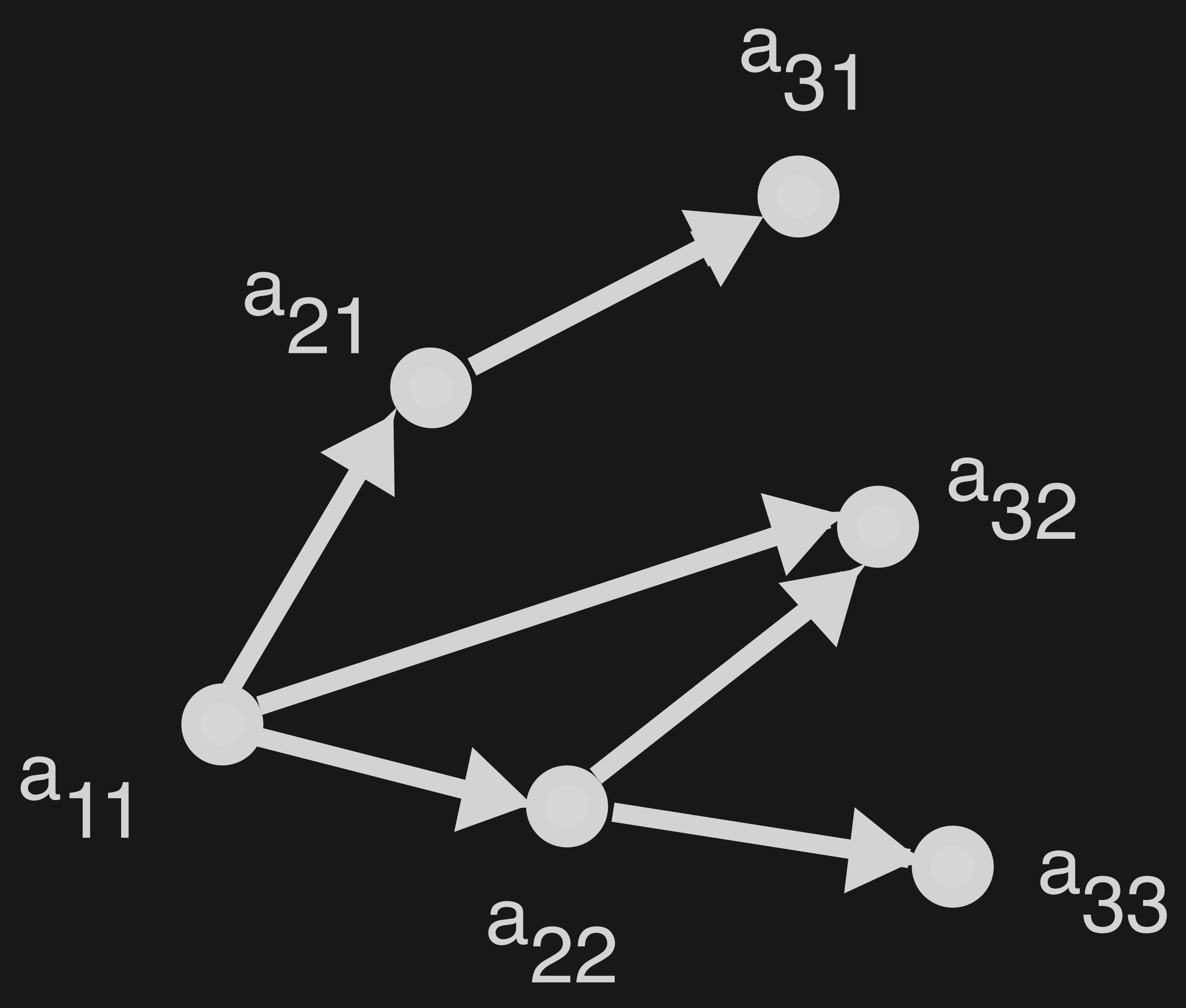
build result tuples by extending prefixes

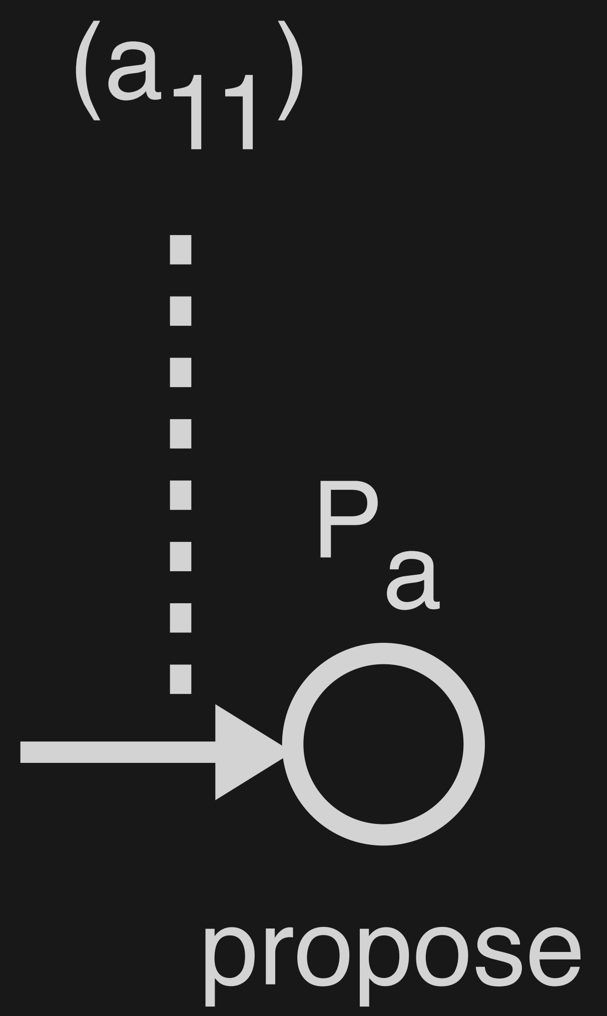
(a_{11})

(a_{11}, a_{22})

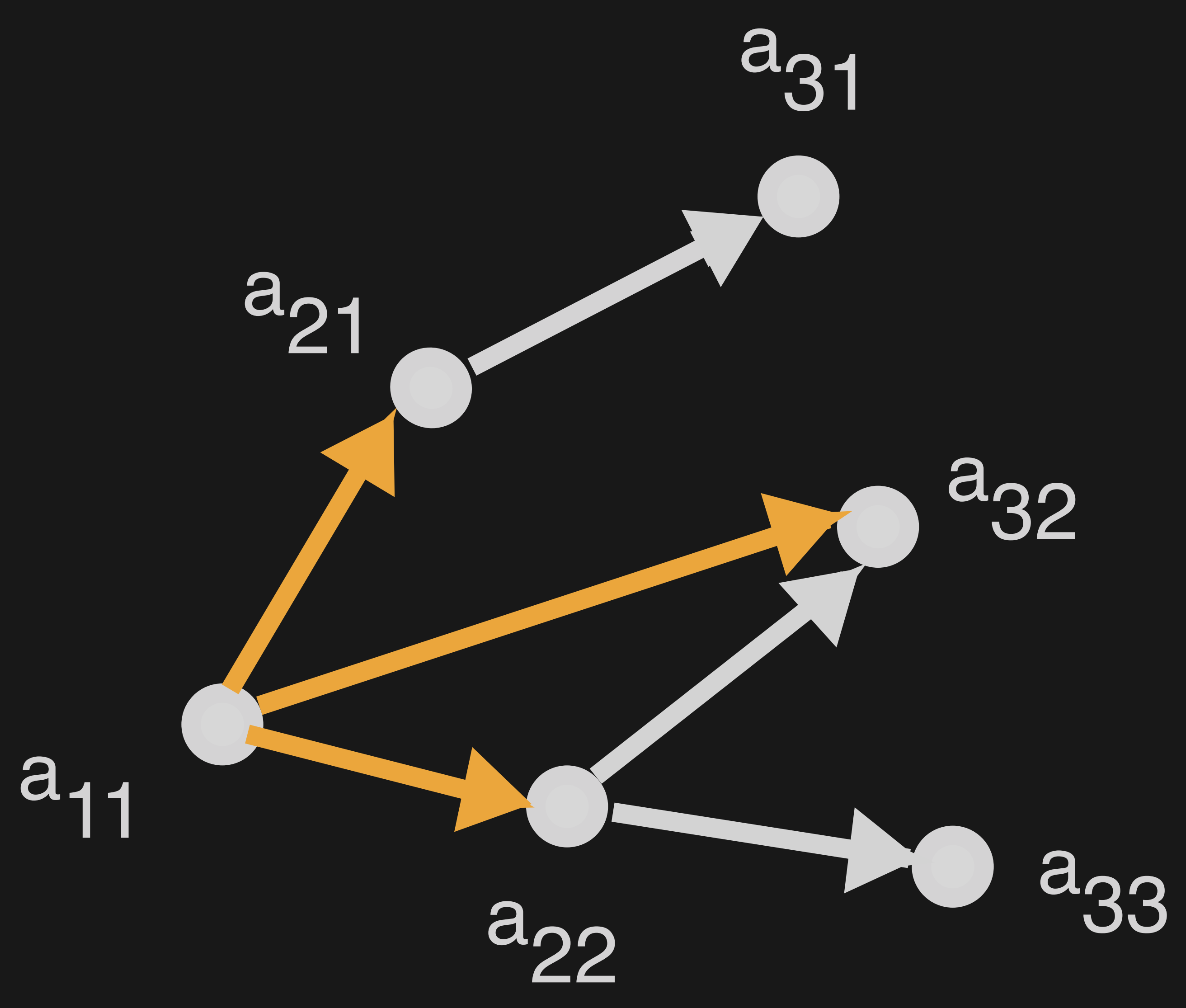
(a_{11}, a_{22}, a_{32})

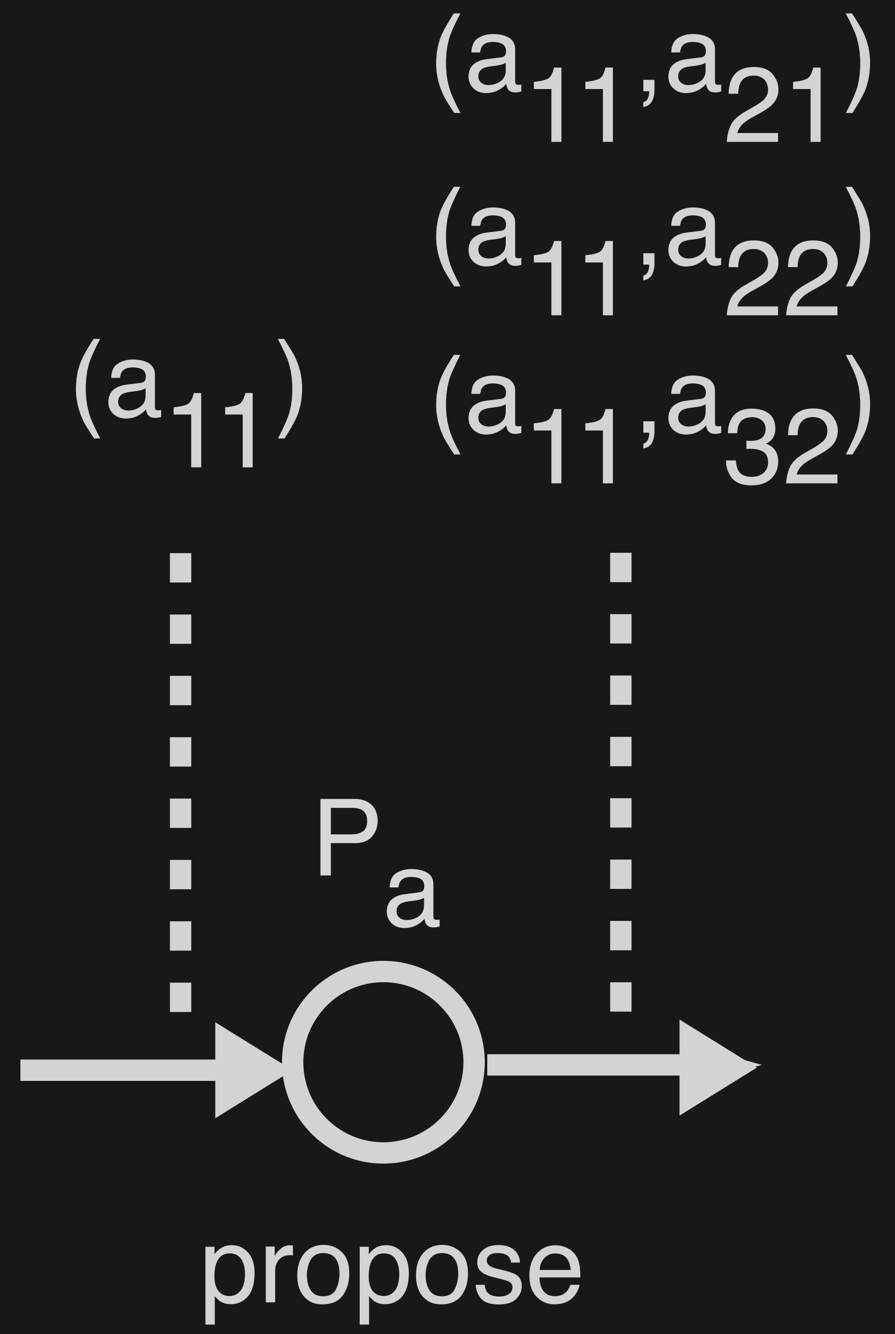
Example - Enumerate triangles in a directed graph



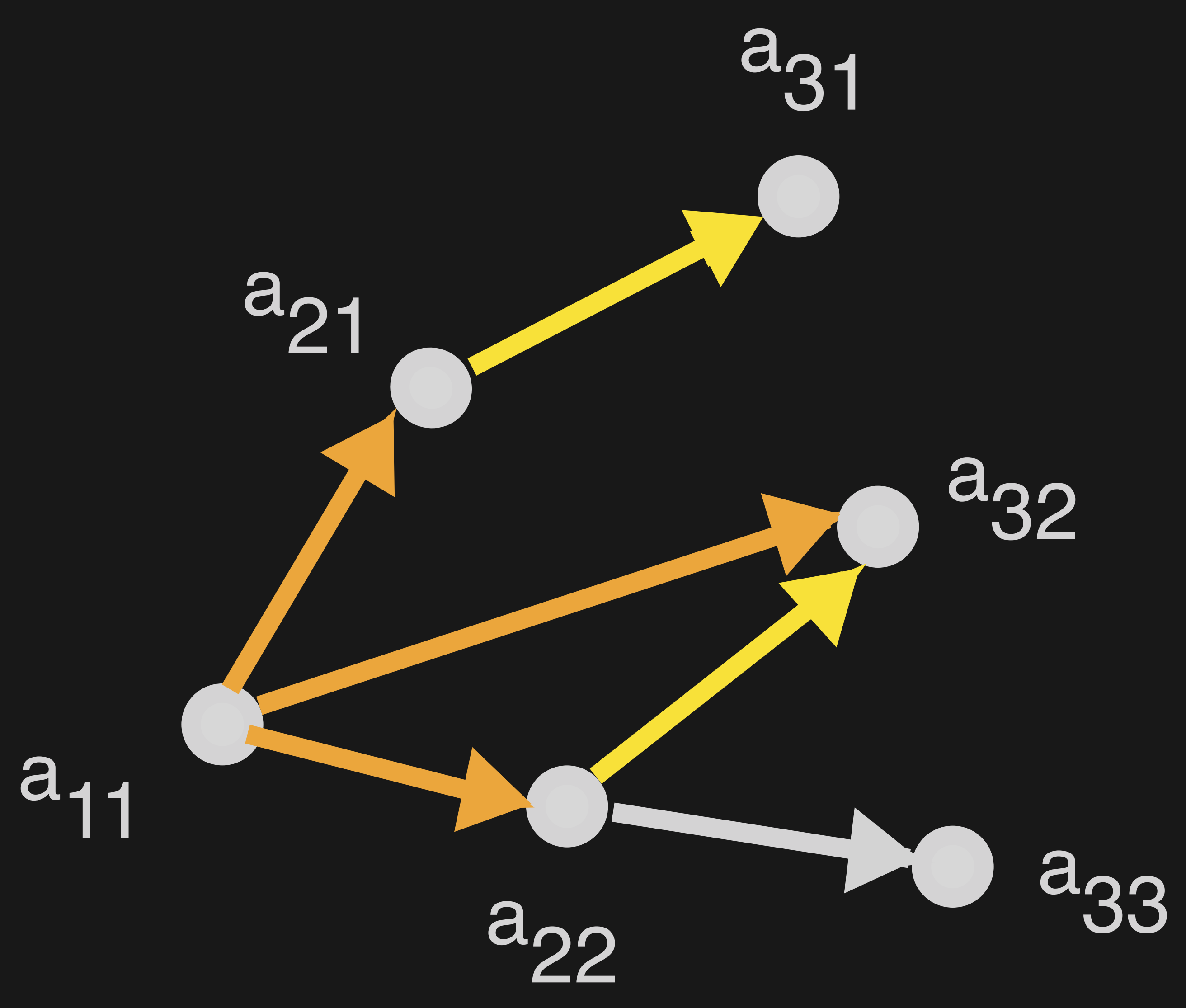


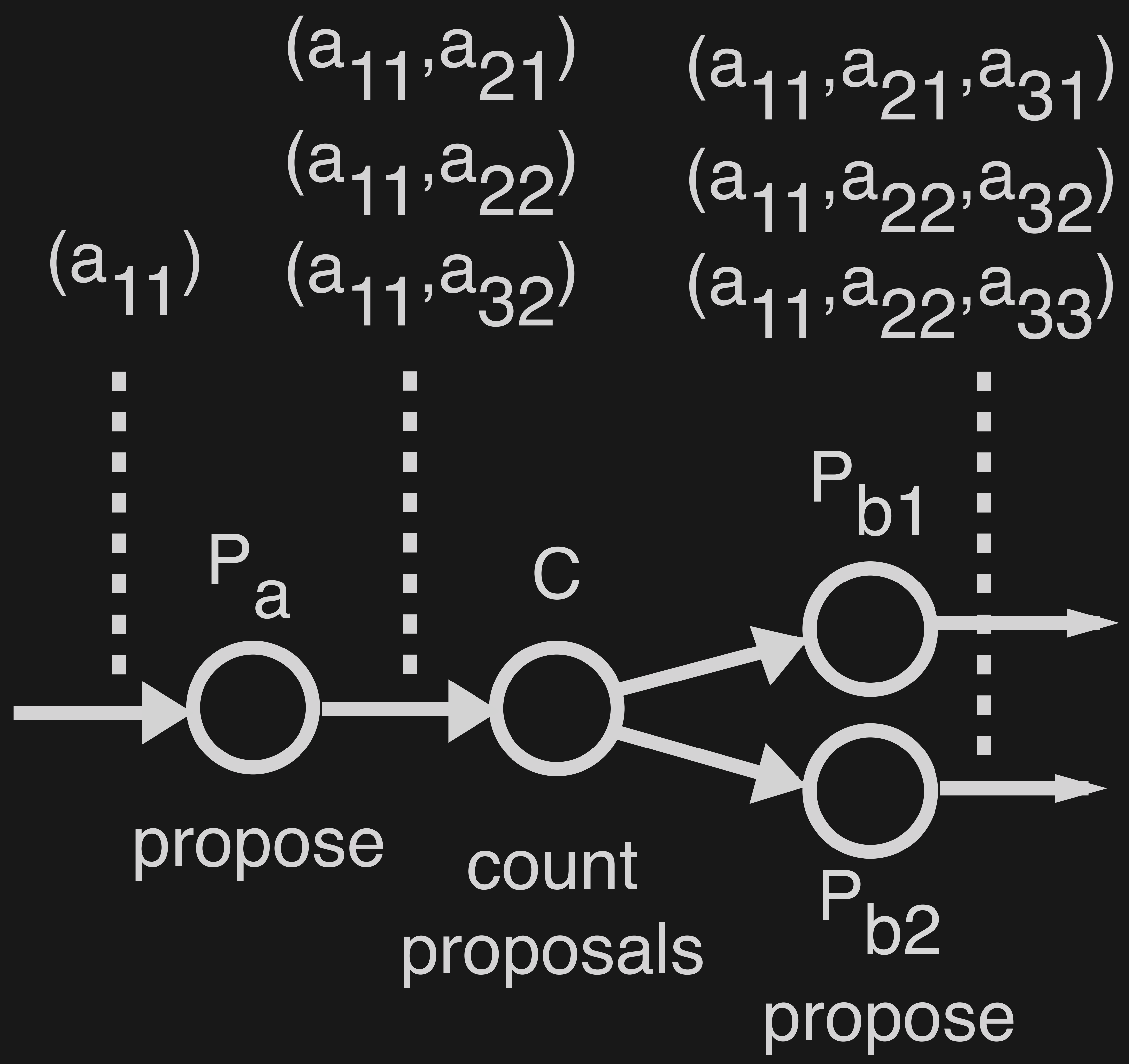
Example - Enumerate triangles in a directed graph



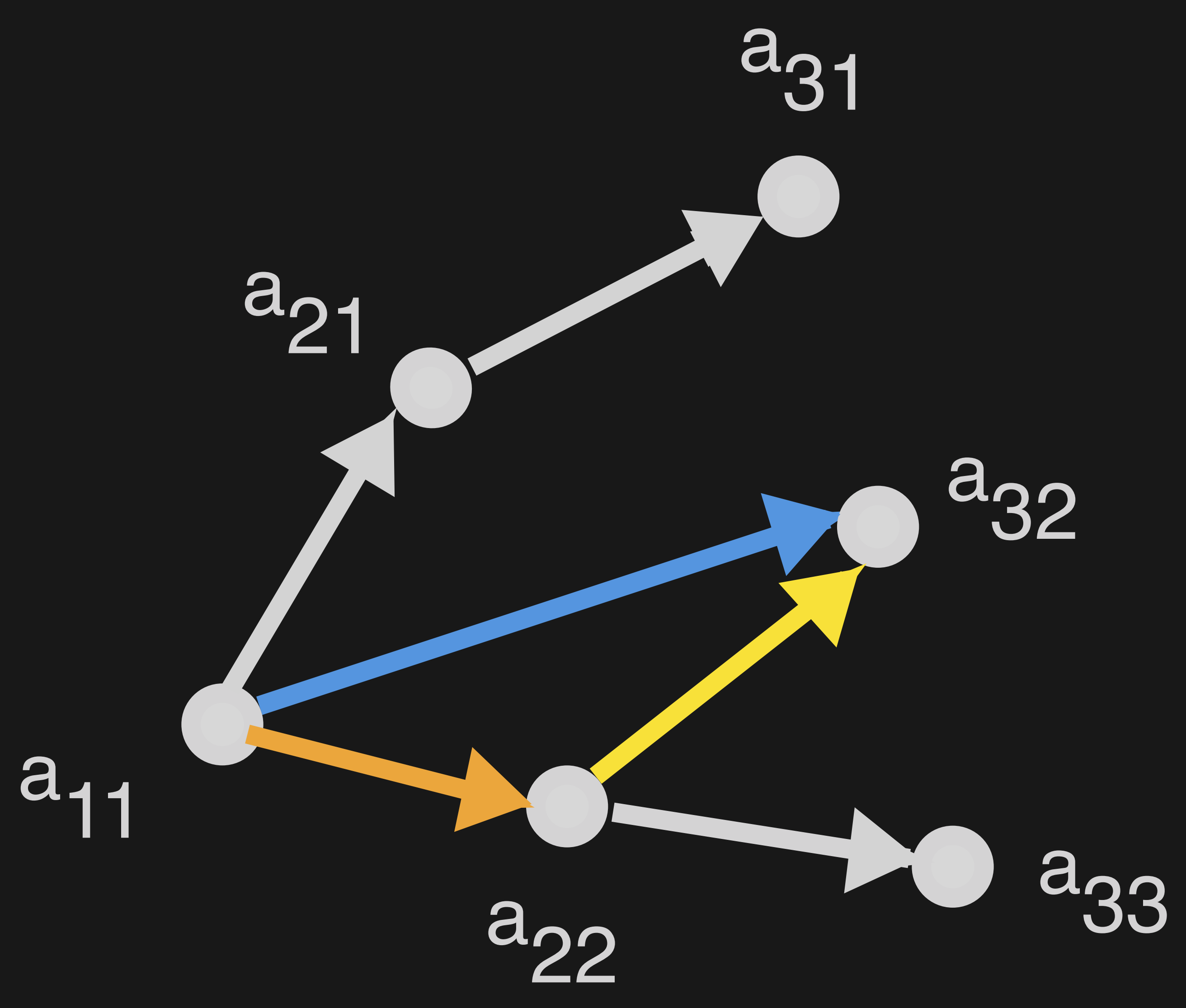


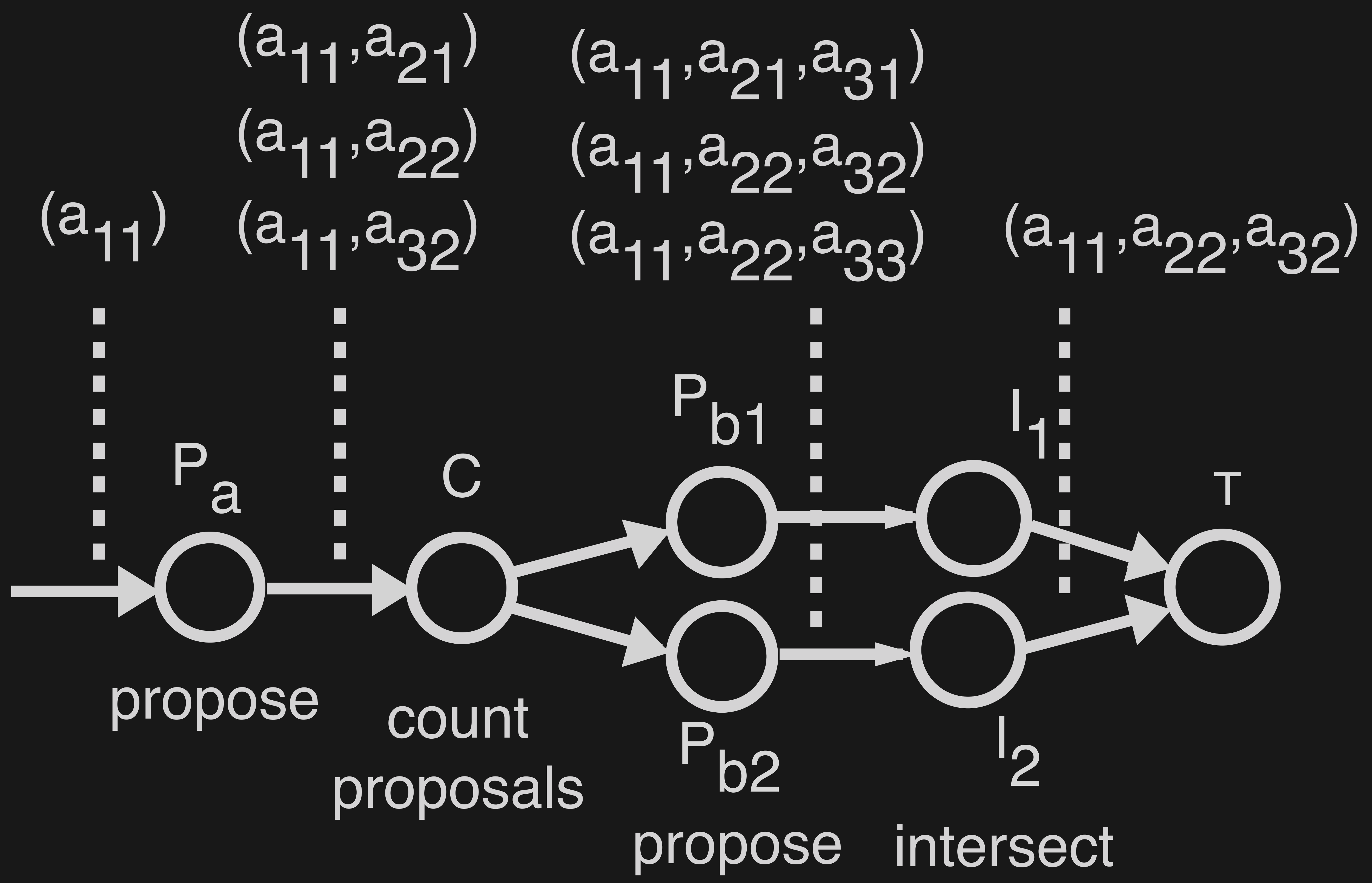
Example - Enumerate triangles in a directed graph



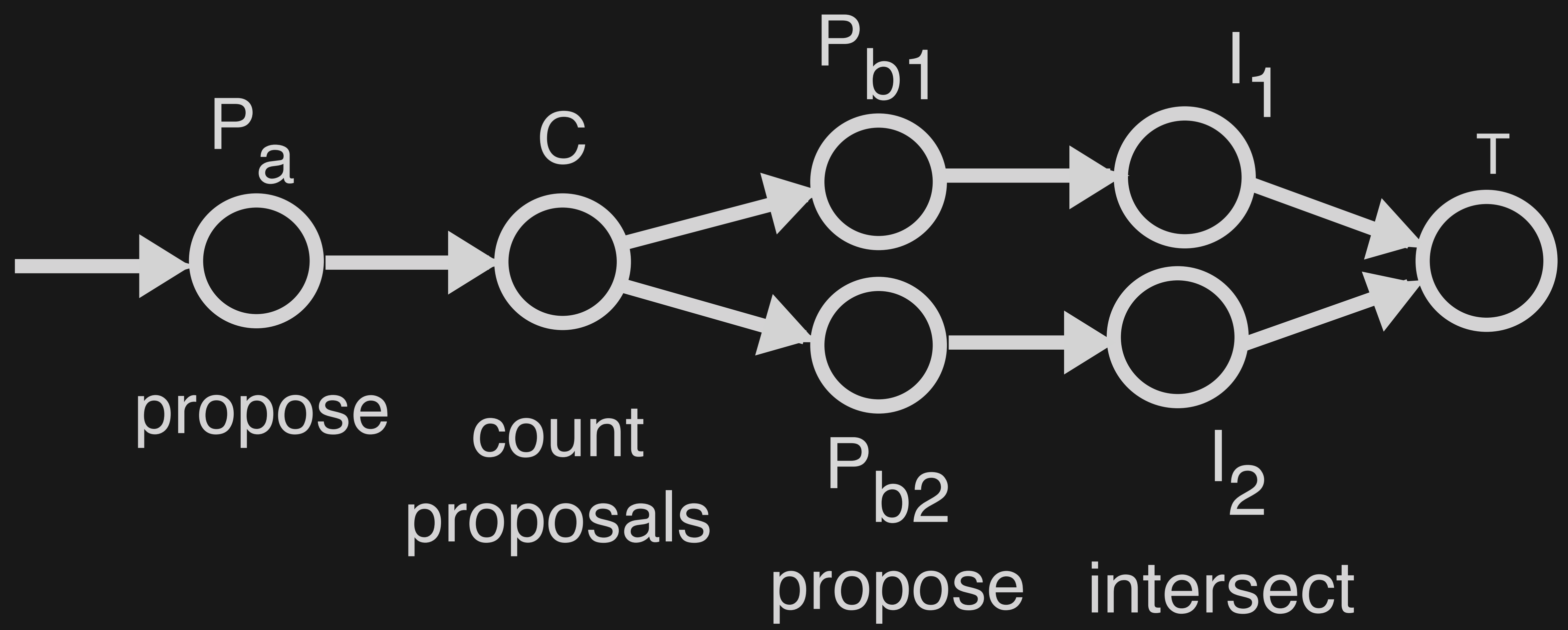


Example - Enumerate triangles in a directed graph

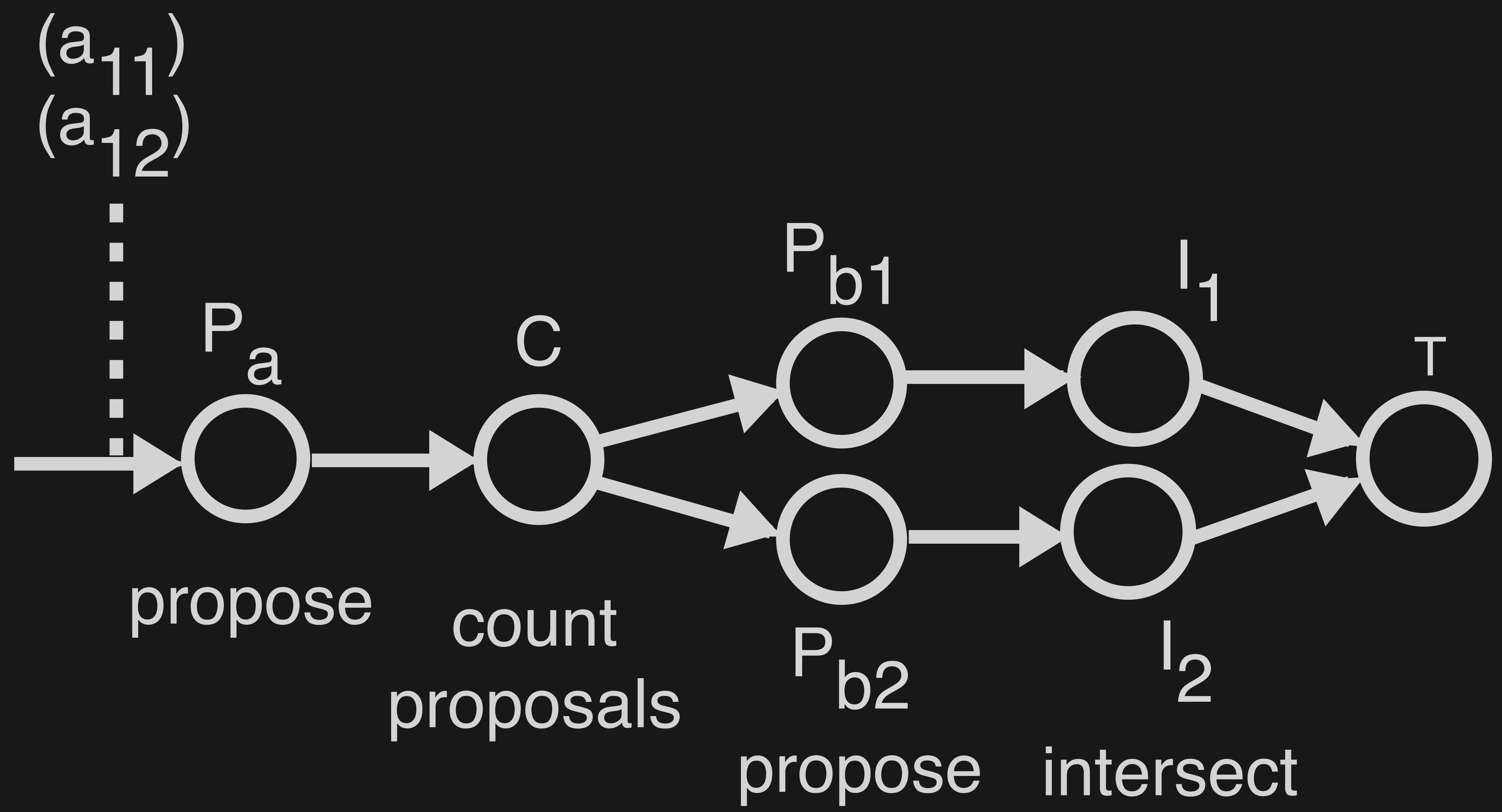




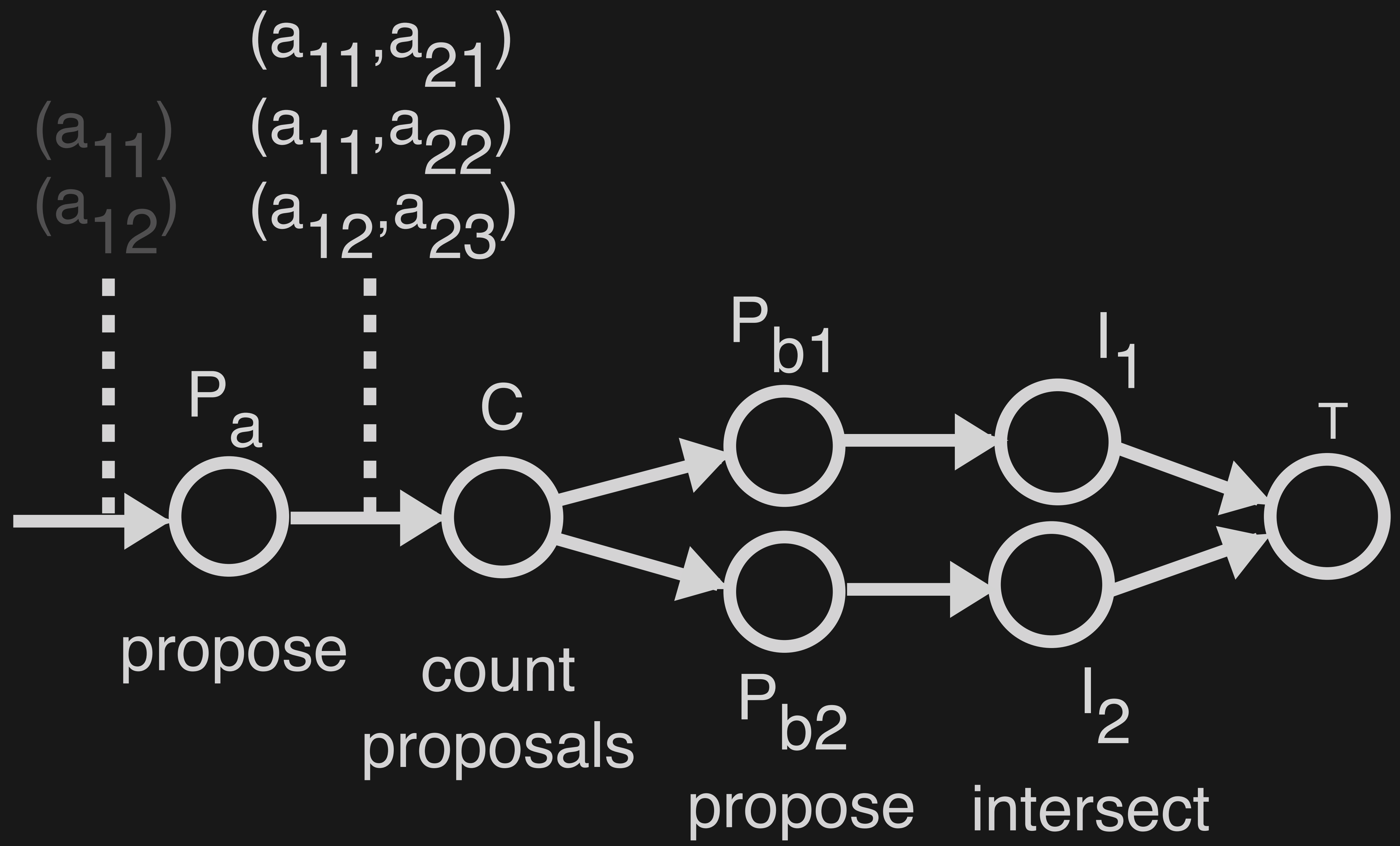
A naïve schedule can generate large *intermediate state*



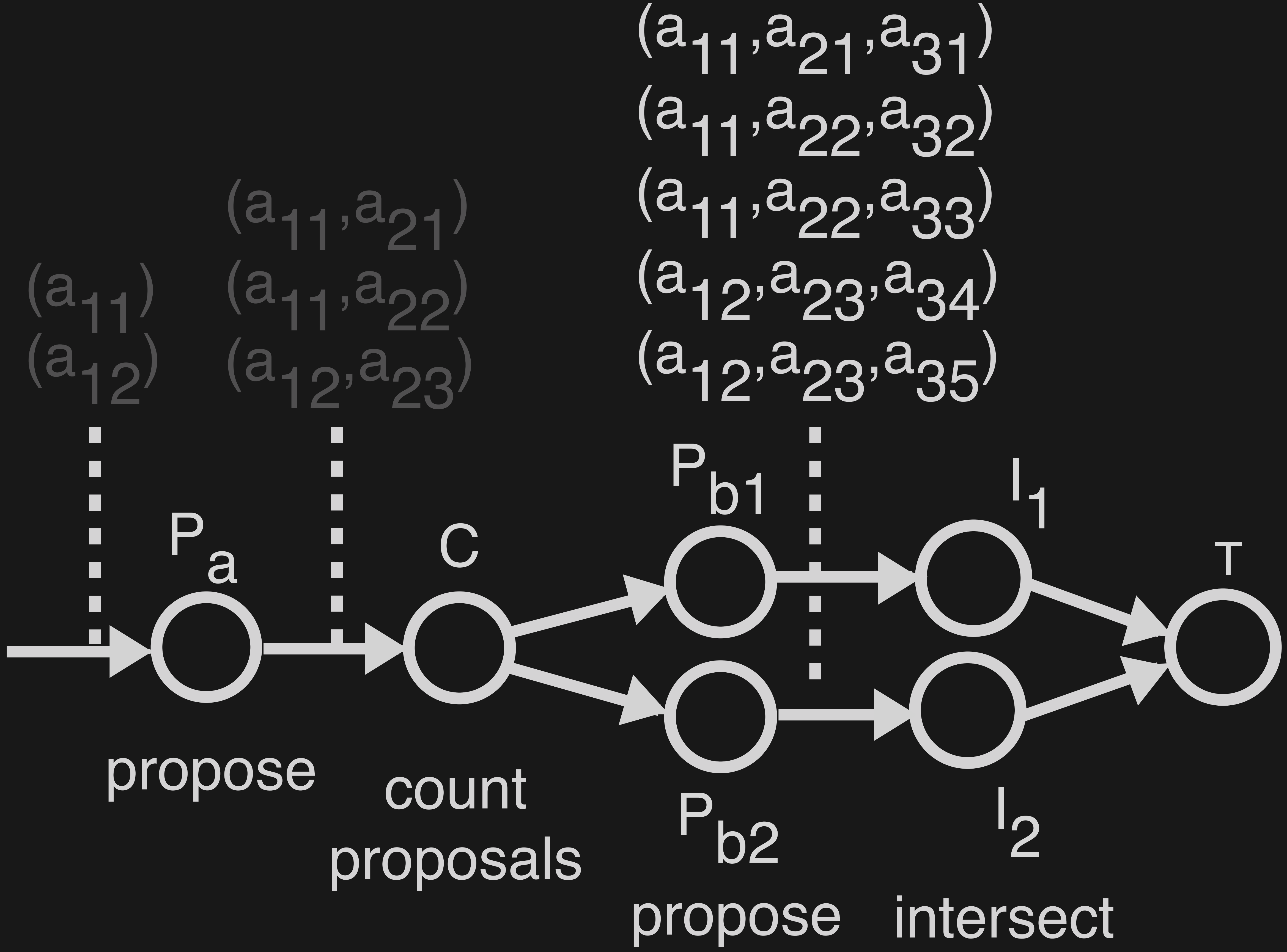
A naïve schedule can generate large *intermediate state*



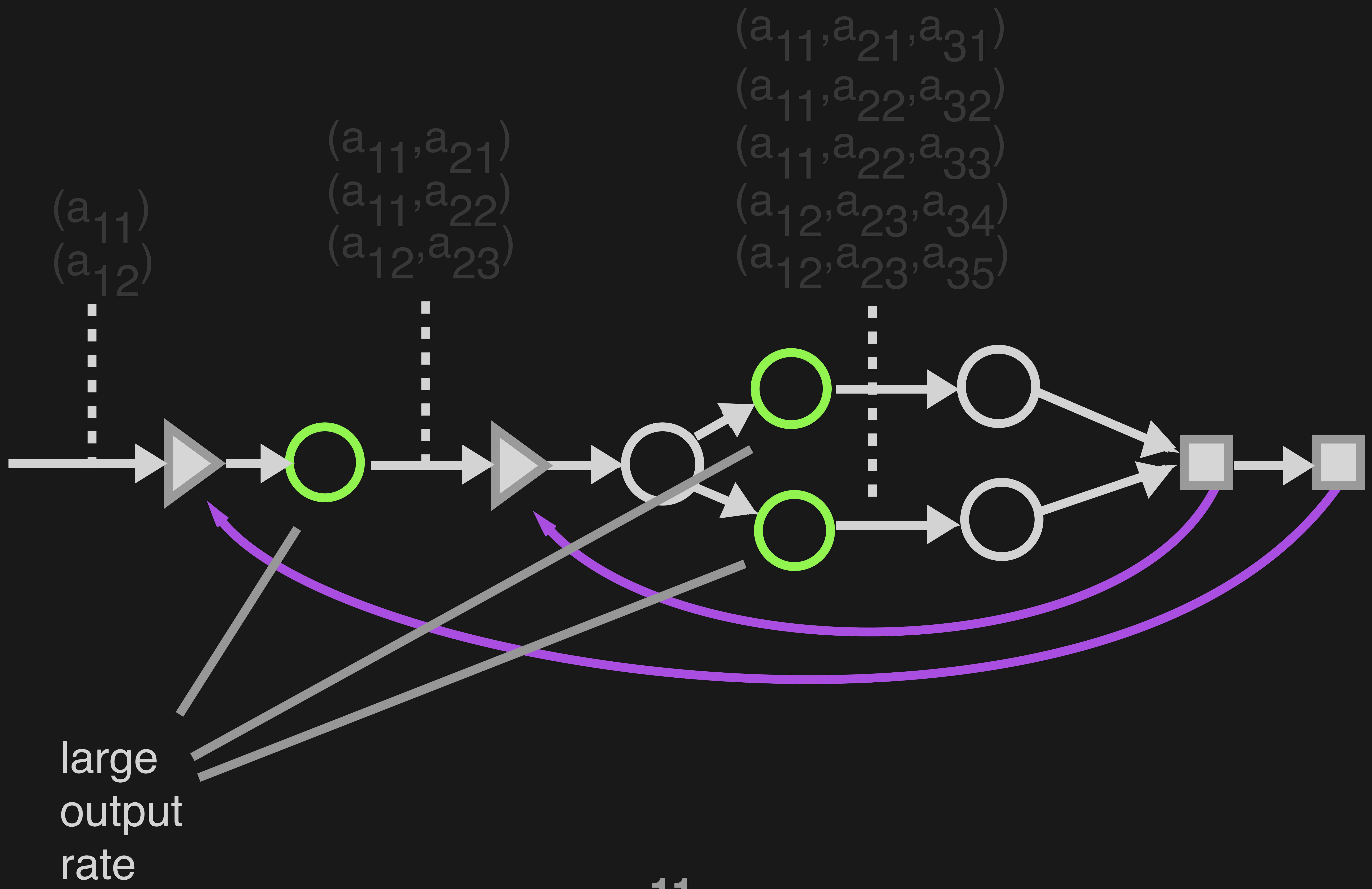
A naïve schedule can generate large *intermediate state*



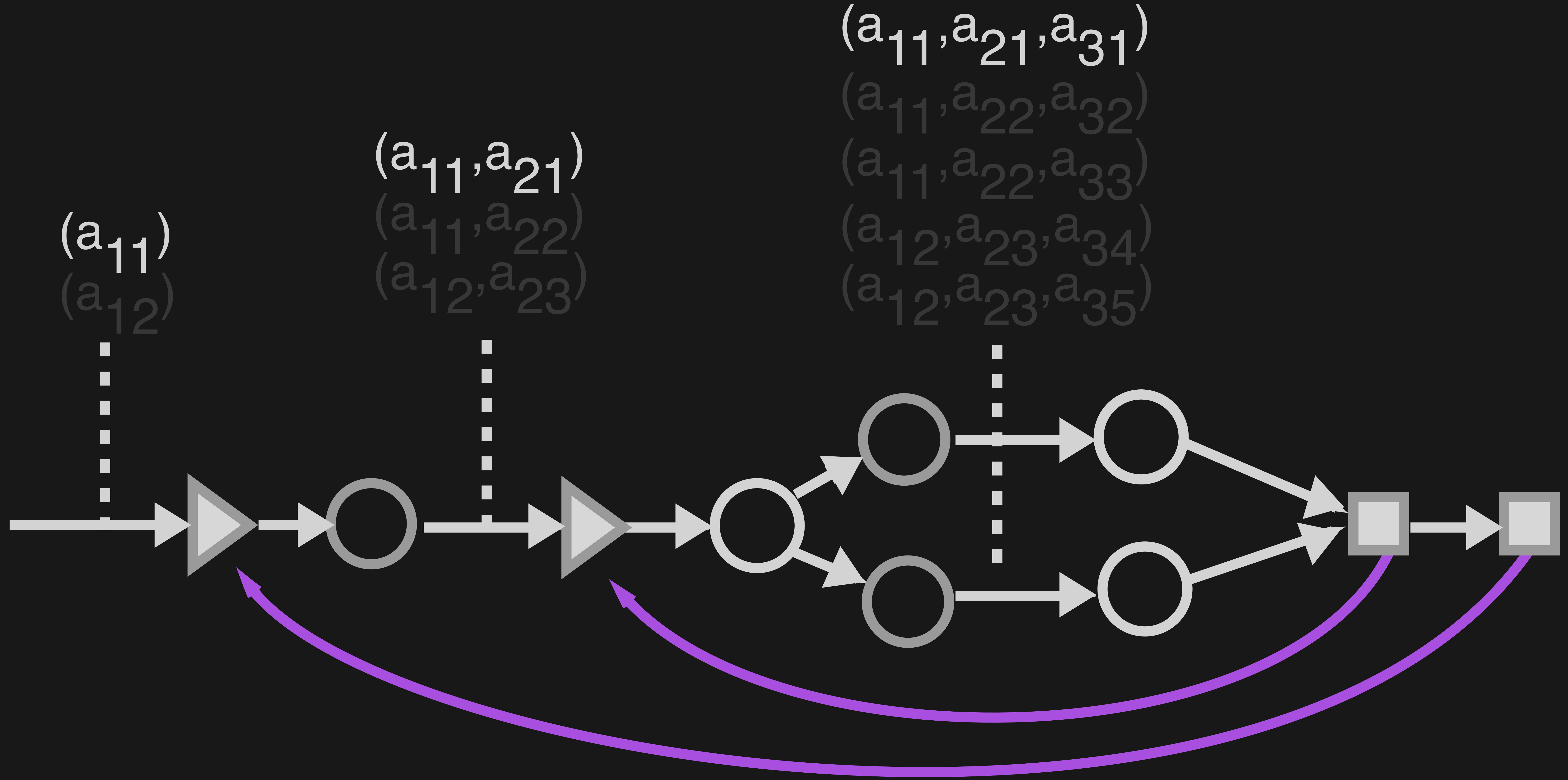
A naïve schedule can generate large *intermediate state*



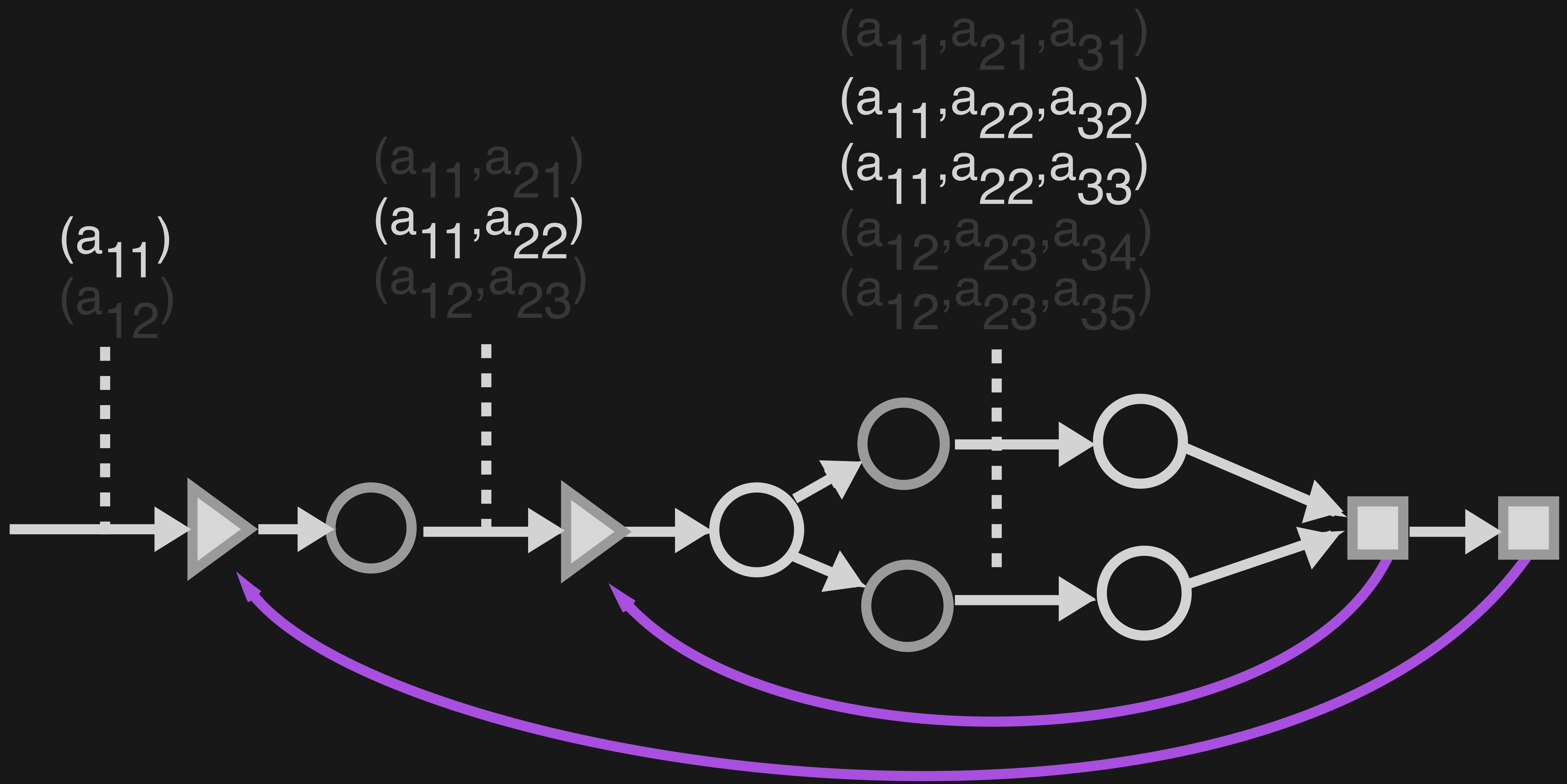
Faucet *limits buffered intermediate results*



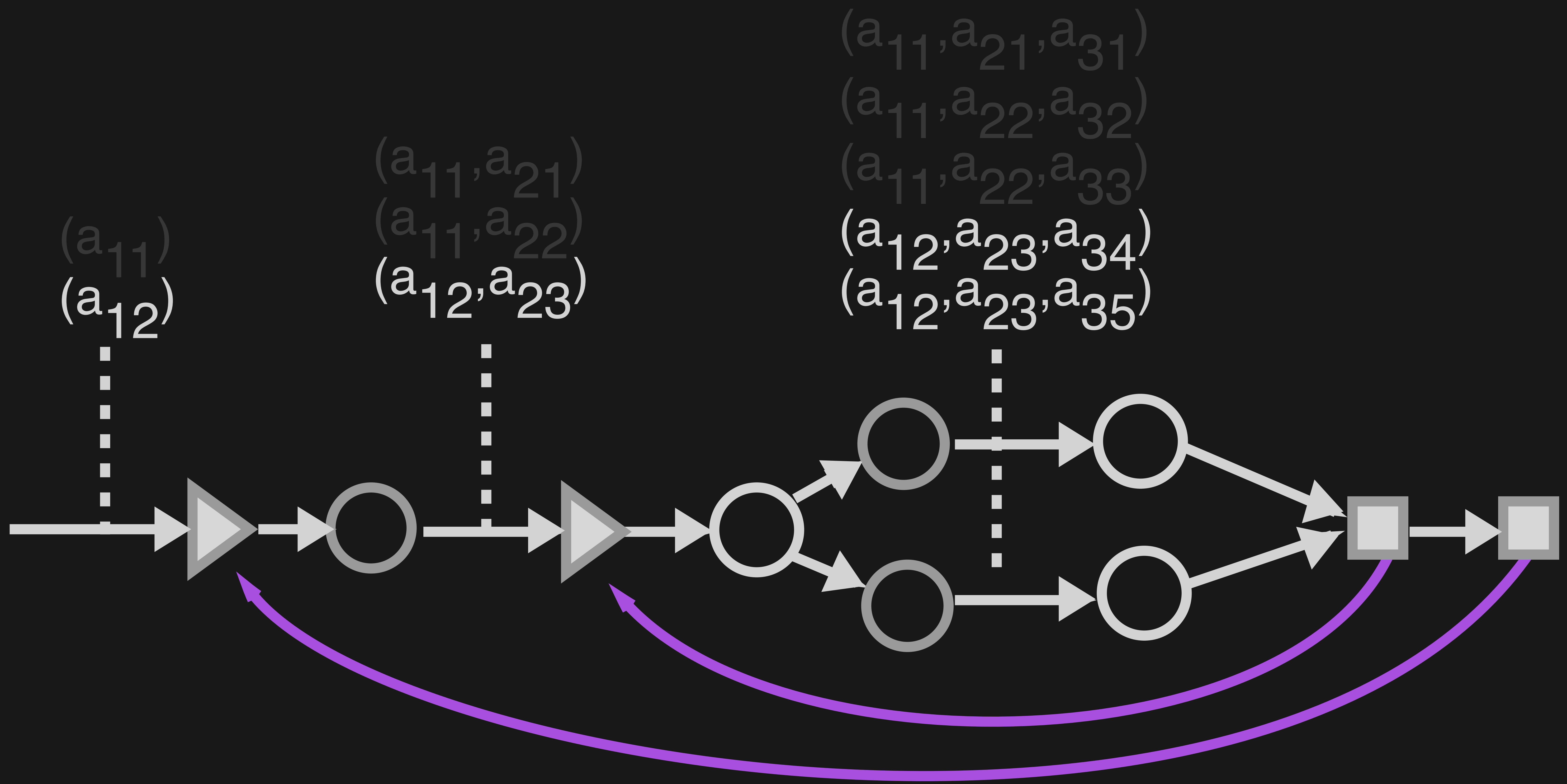
Faucet *limits buffered intermediate results*



Faucet *limits buffered intermediate results*



Faucet *limits buffered intermediate results*


(a_{11})
(a_{12})
(a_{11},a_{21})
(a_{11},a_{22})
(a_{12},a_{23})
(a_{11},a_{21},a_{31})
(a_{11},a_{22},a_{32})
(a_{11},a_{22},a_{33})
(a_{12},a_{23},a_{34})
(a_{12},a_{23},a_{35})

Evaluation - Dataset

Enumerate triangles in the **Livejournal Dataset**

4’847’571 nodes

68’993’773 edges

285’730’264 triangles

Hardware

Intel Xeon E5-2650 @ 2.00GHz

16 physical cores

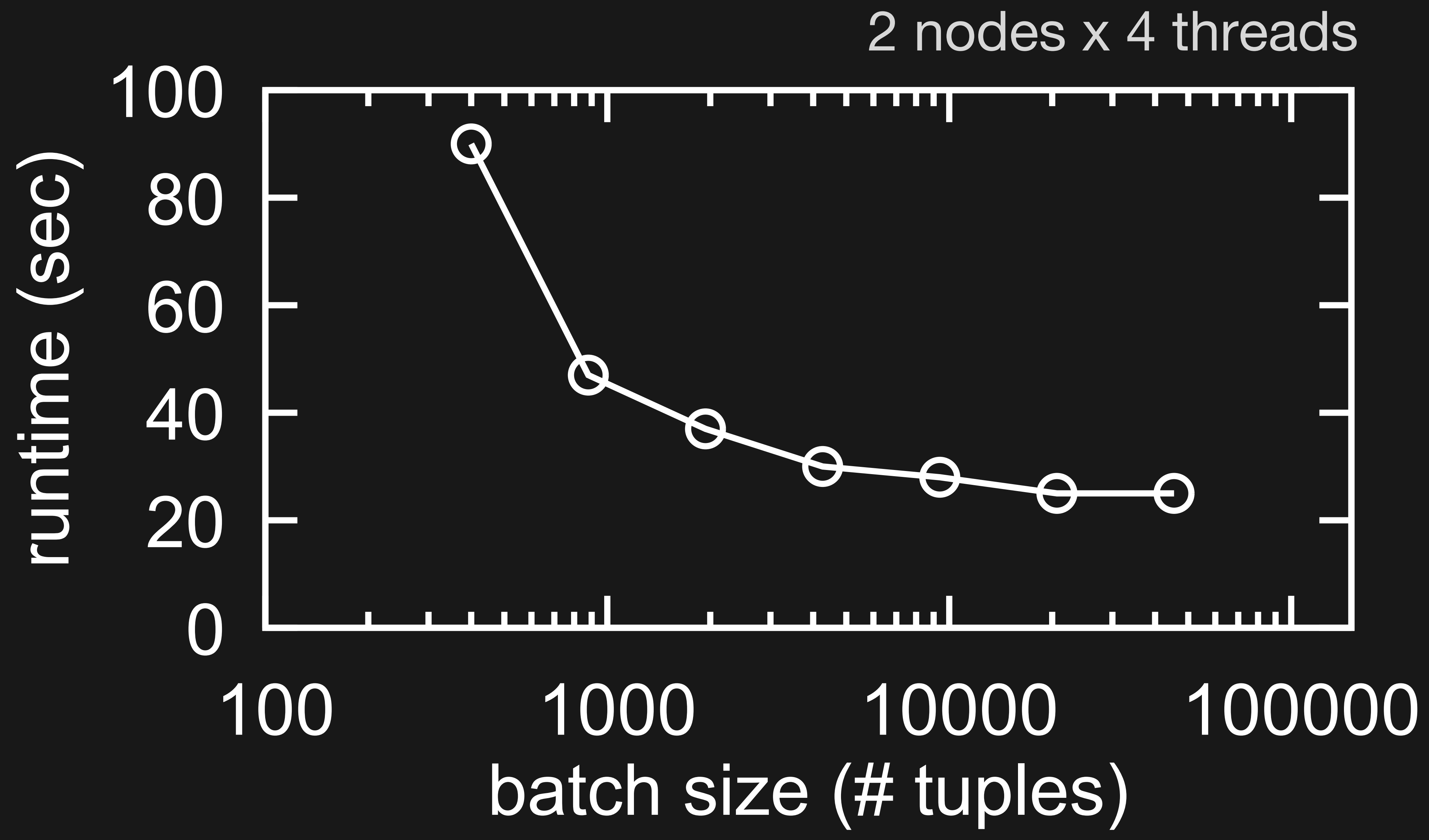
10Gbps link

Evaluation - Sensitivity to parameter choice

N_{batches} number of batches in-flight in parallel

$N_{\text{batches}} \geq 2$ mitigates stragglers

B batch size

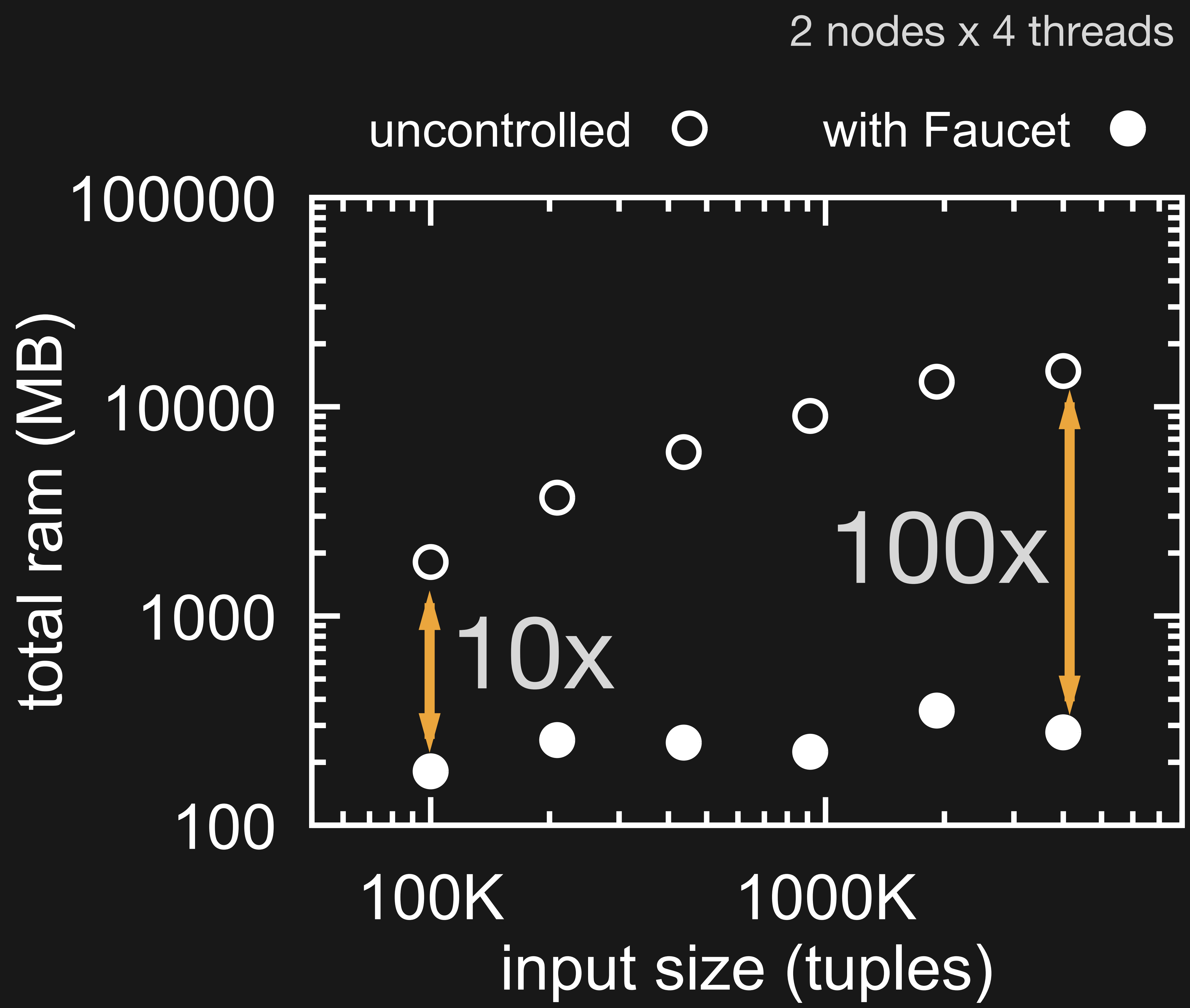


Evaluation

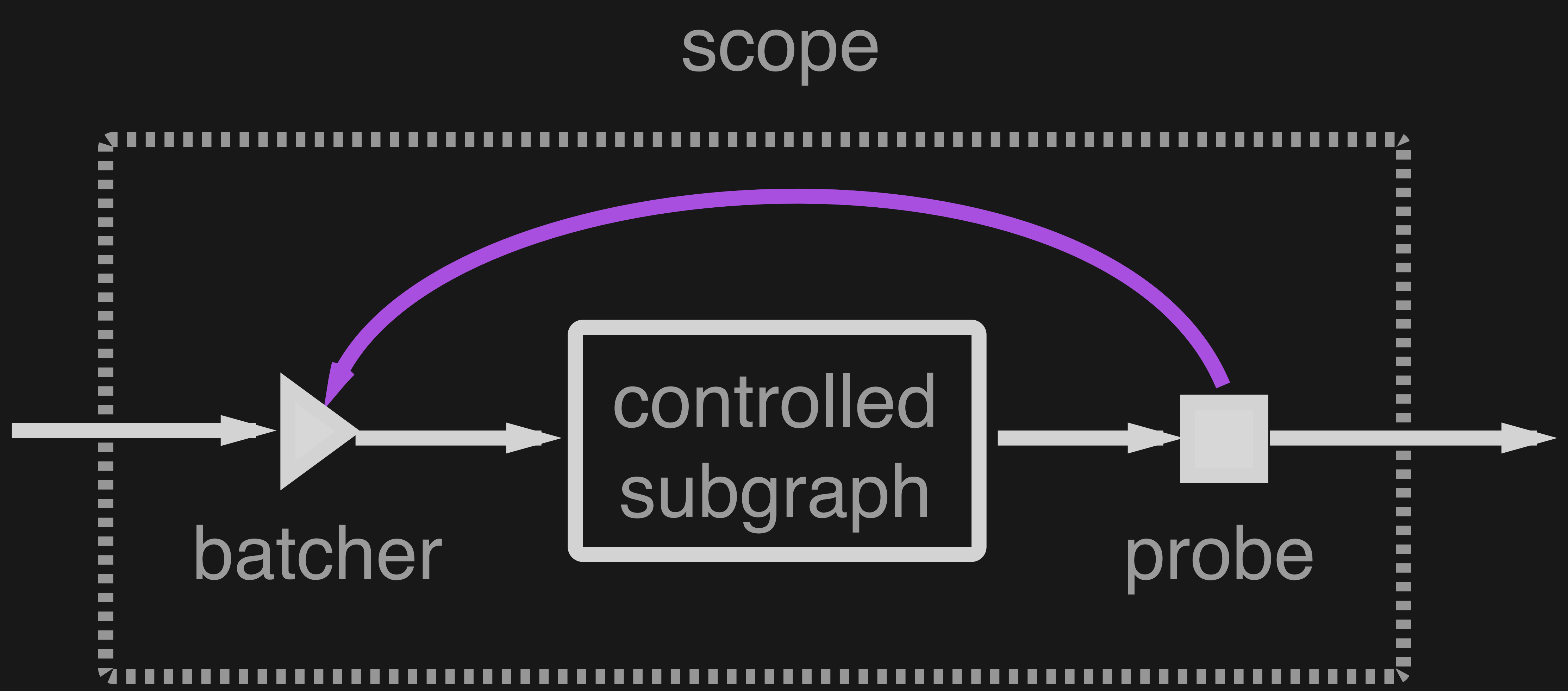
Runtime overhead

15-25%

Memory savings



Faucet



limits intermediate state

RAM is increasingly the main cost of a system

Memory savings	10-100x or more
Overhead	15-25%