UNIVERSITY OF COPENHAGEN

ETH zürich

Verified Progress Tracking for Timely Dataflow

Matthias Brun, Sára Decova, Andrea Lattuada
Department of Computer Science, ETH Zürich

Dmitriy Traytel
Department of Computer Science, University of Copenhagen

UNIVERSITY OF COPENHAGEN

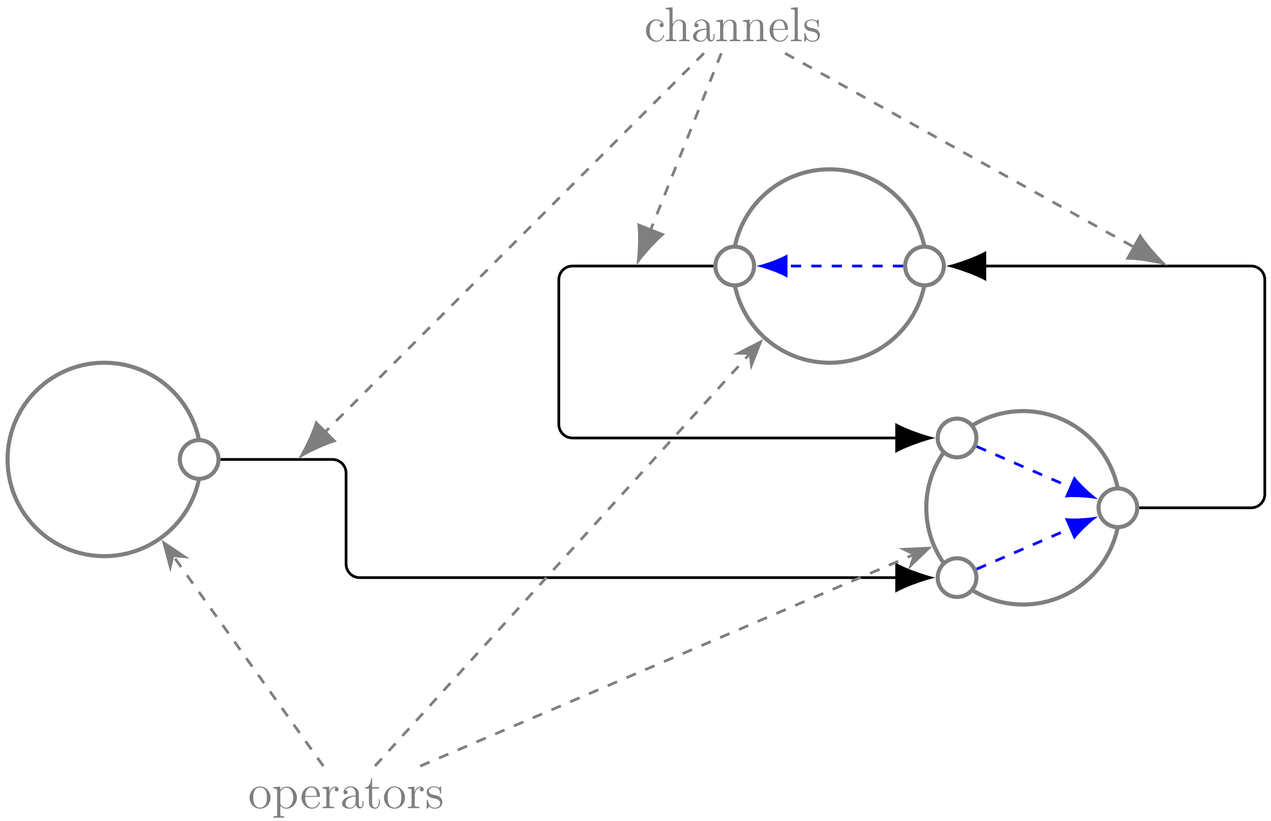
ETH zürich

Verified Progress Tracking for Timely Dataflow

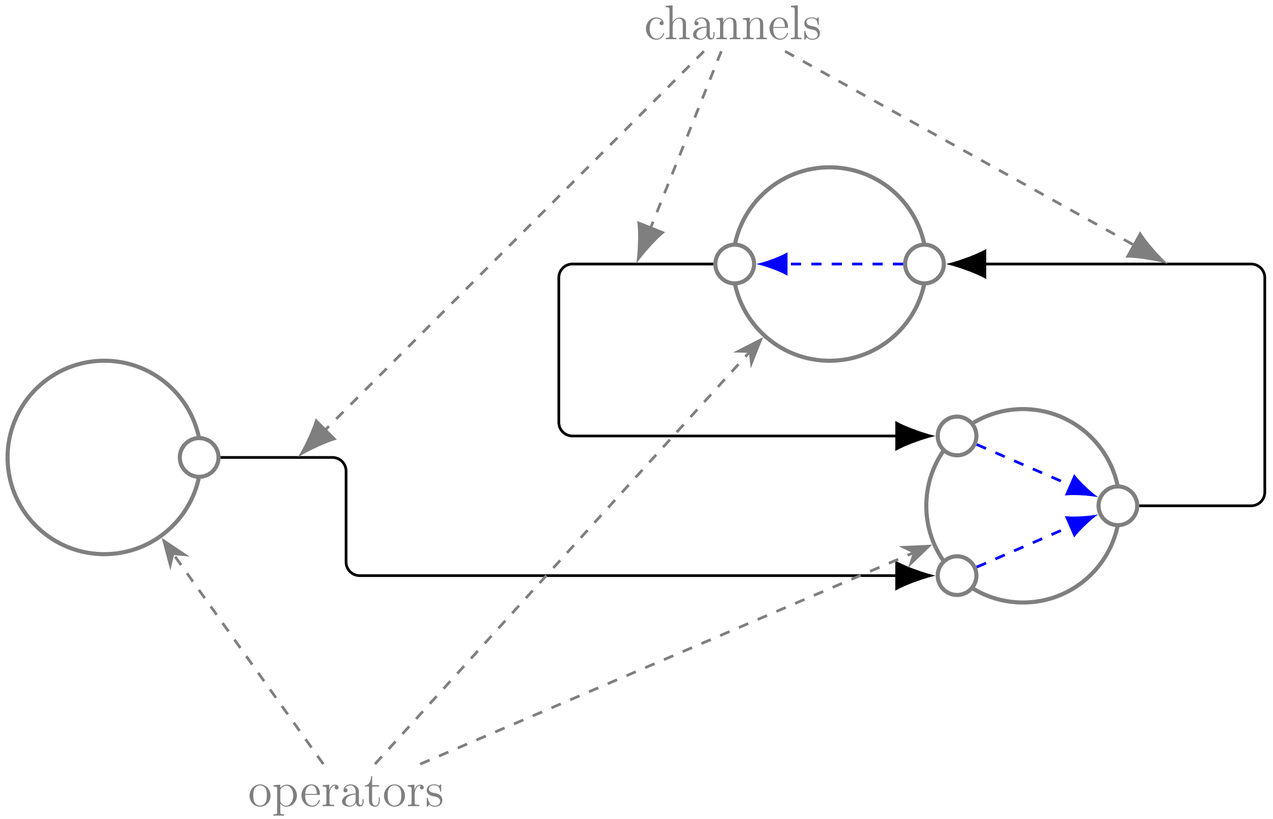
Matthias Brun, Sára Decova, Andrea Lattuada
Department of Computer Science, ETH Zürich

Dmitriy Traytel
Department of Computer Science, University of Copenhagen

A dataflow program



A dataflow program



Timely Dataflow

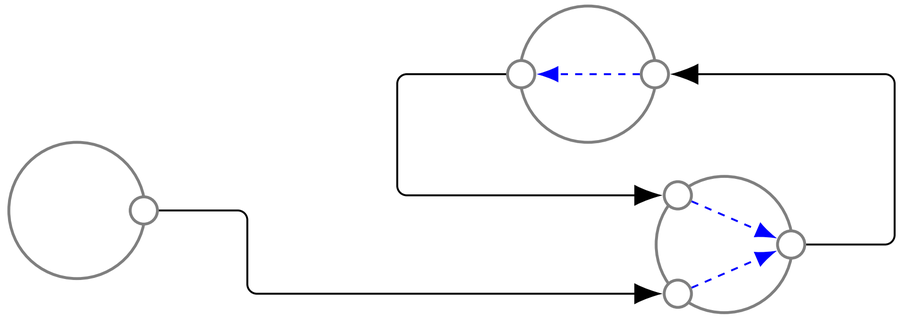
Time-aware dataflow system

Each tuple (message) tagged with timestamp

↳ logical/temporal grouping

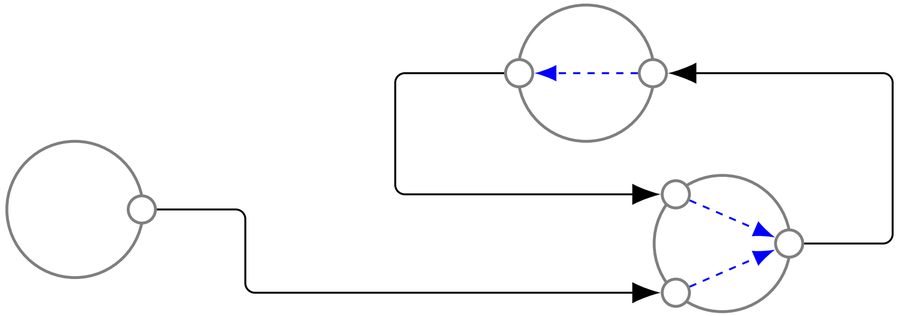
Dataflow programming for automatic parallelzation

logical dataflow

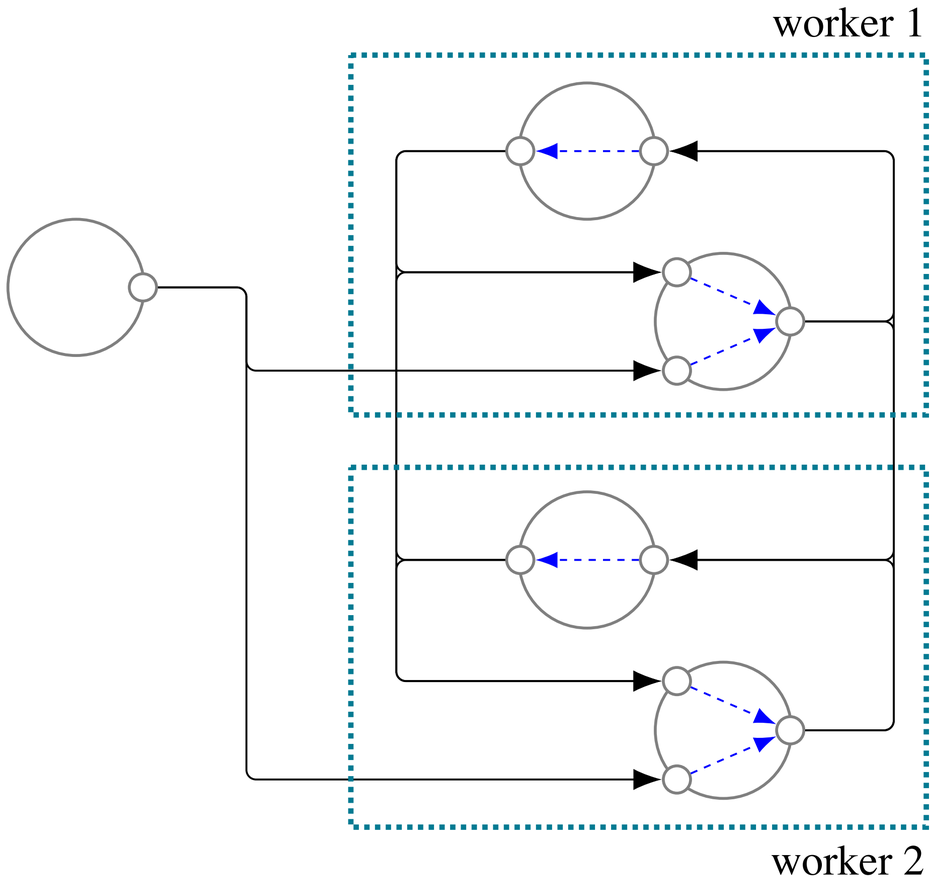


Dataflow programming for automatic parallelzation

logical dataflow

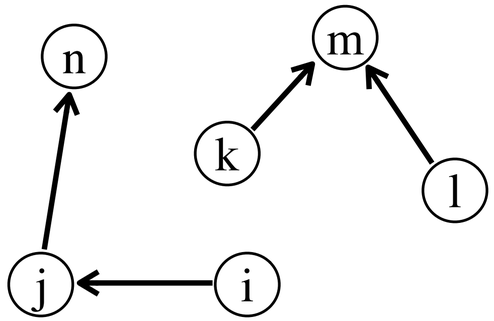


physical dataflow



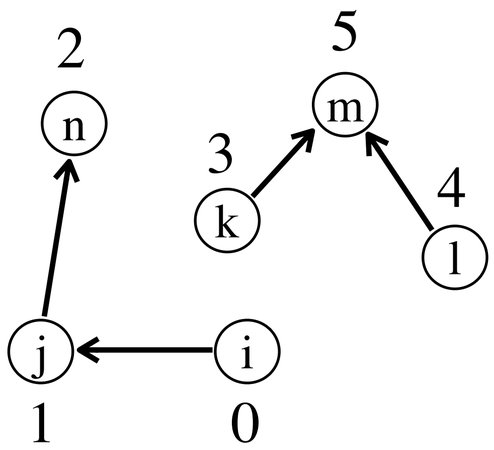
WCC by propagating labels

iterative graph algorithm



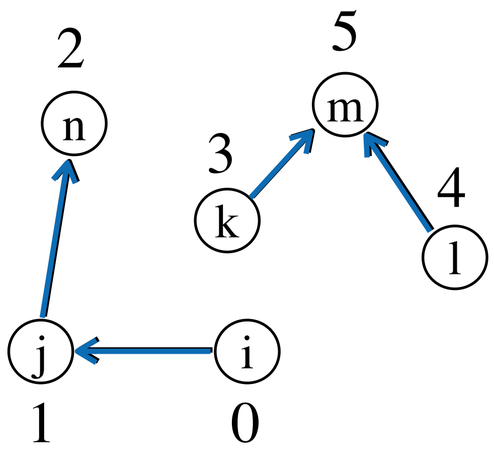
WCC by propagating labels

iterative graph algorithm



WCC by propagating labels

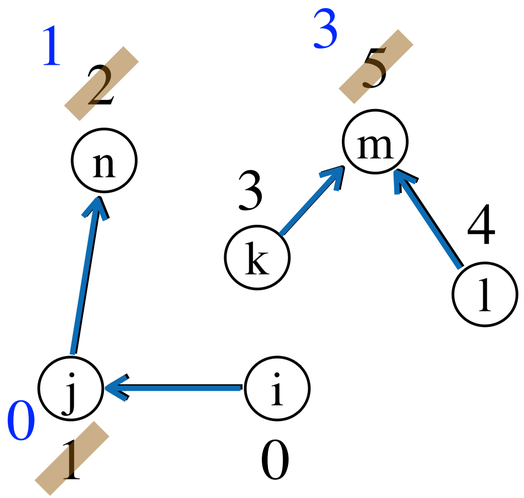
iterative graph algorithm



first propagation

WCC by propagating labels

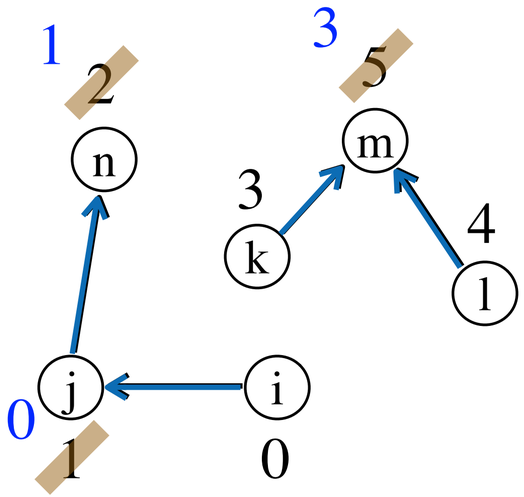
iterative graph algorithm



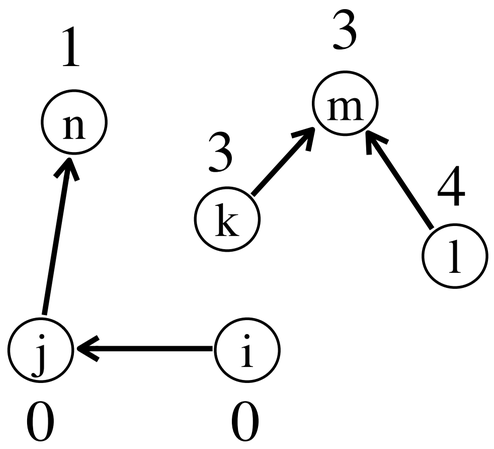
first propagation

WCC by propagating labels

iterative graph algorithm

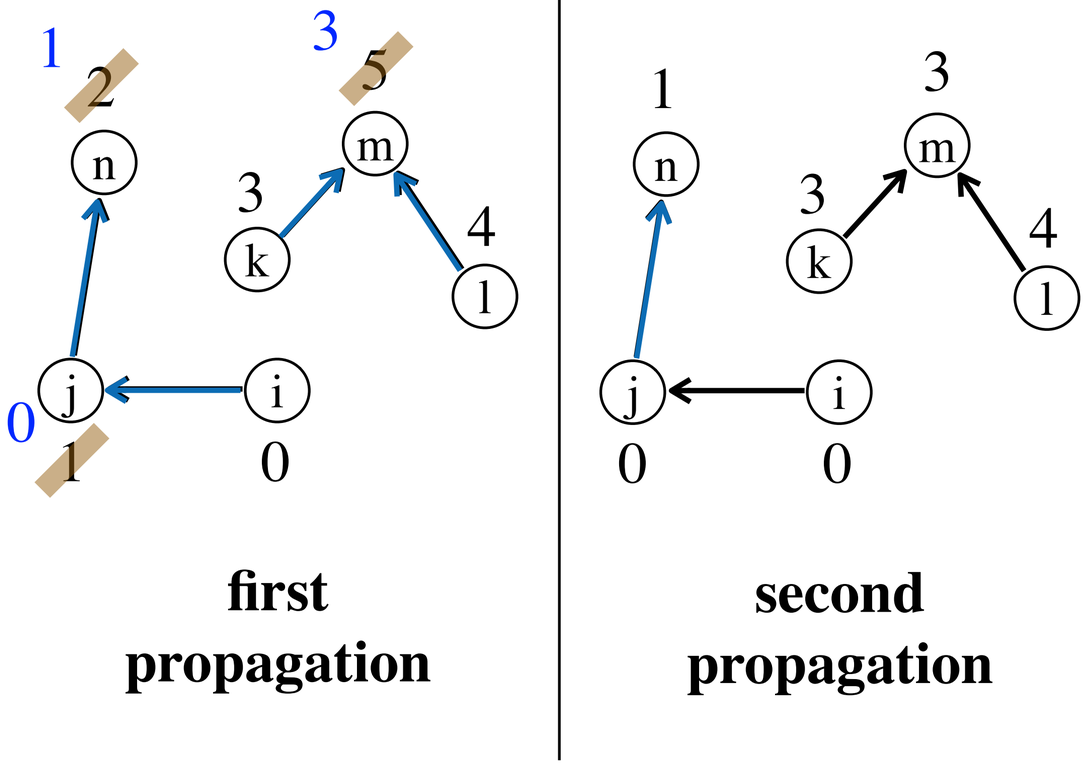


first propagation



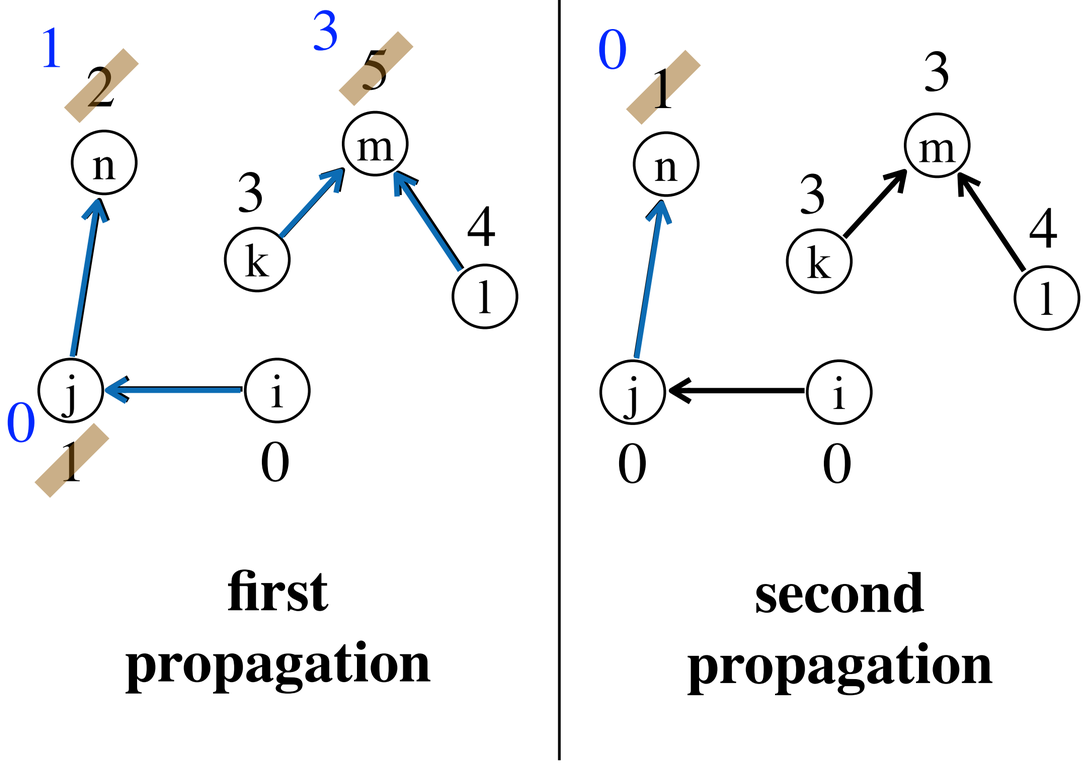
WCC by propagating labels

iterative graph algorithm



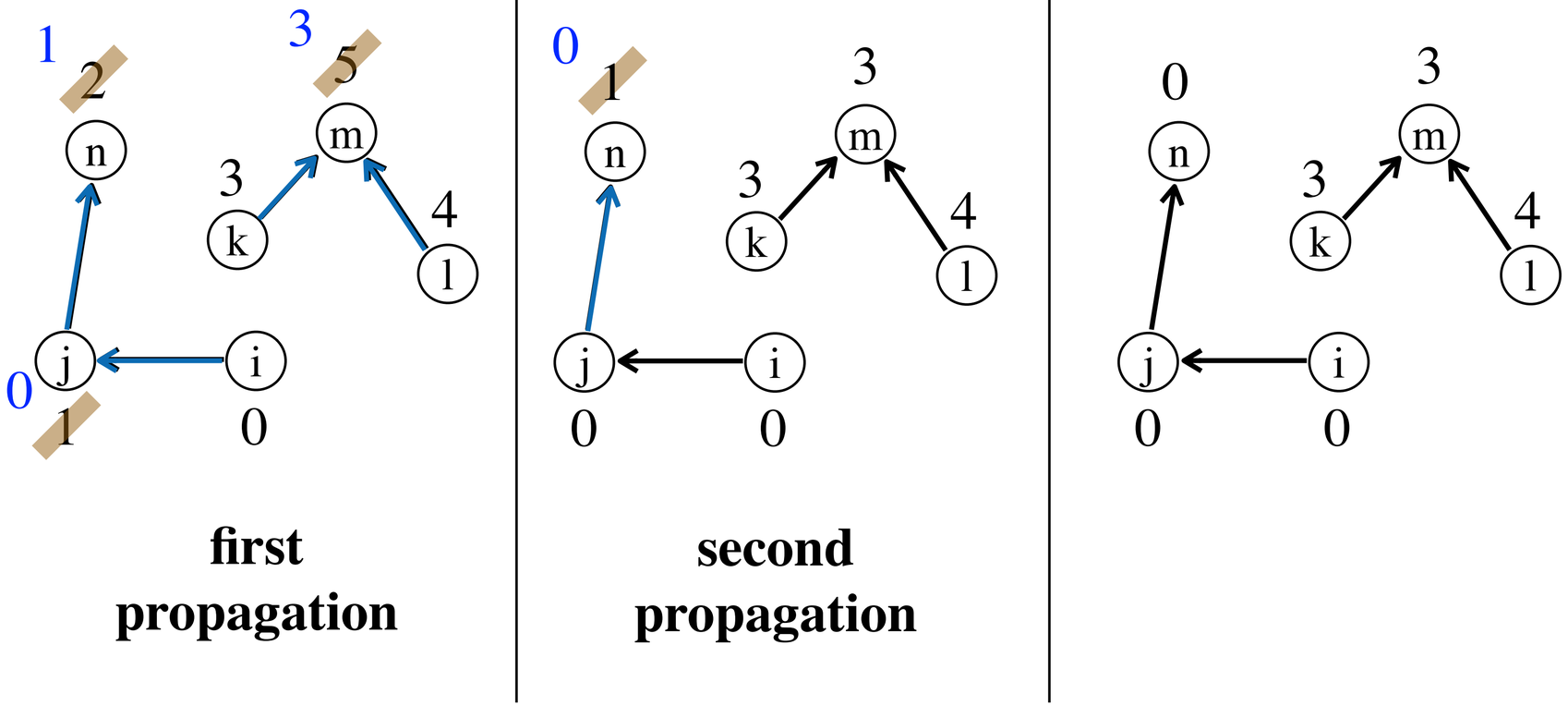
WCC by propagating labels

iterative graph algorithm



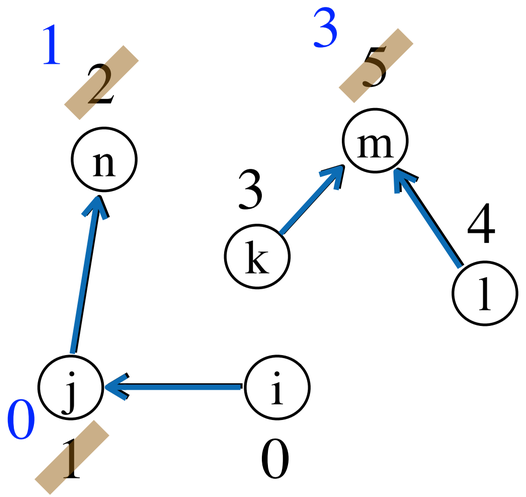
WCC by propagating labels

iterative graph algorithm

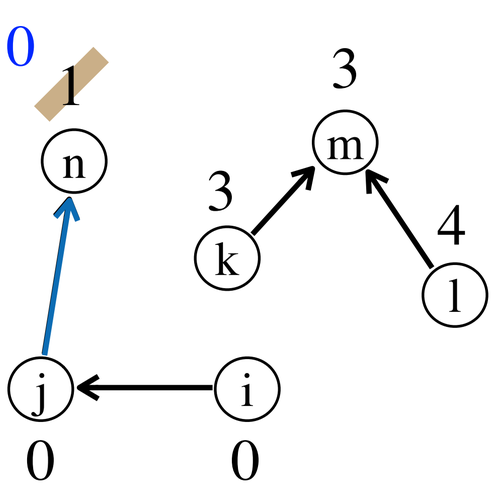


WCC by propagating labels

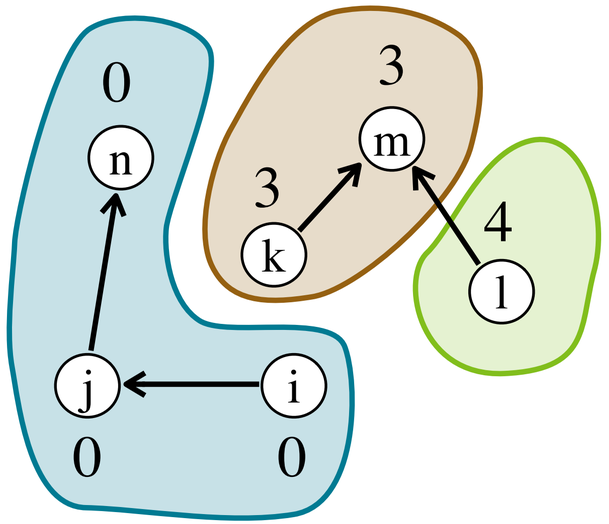
iterative graph algorithm



first propagation

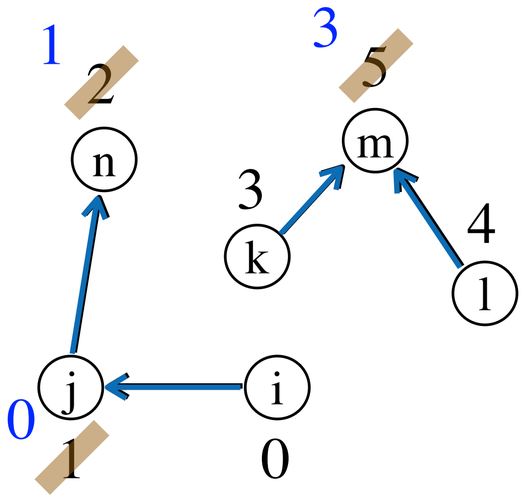


second propagation

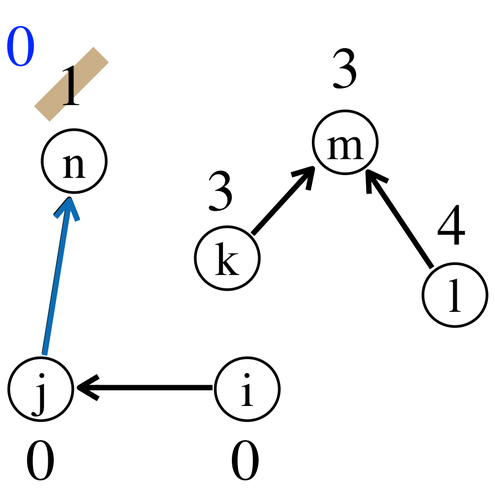


WCC by propagating labels

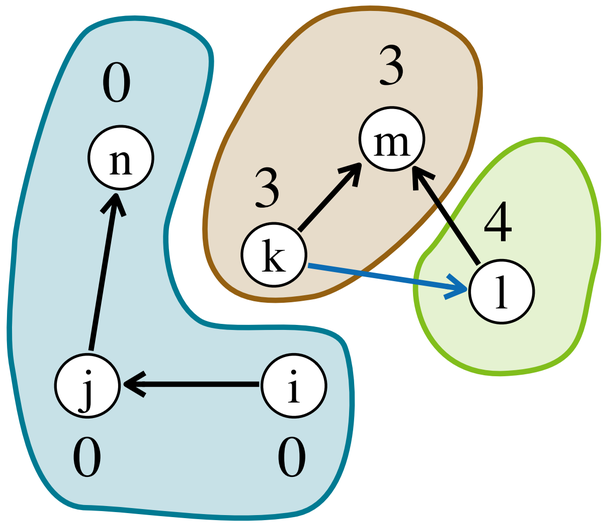
iterative graph algorithm



first propagation



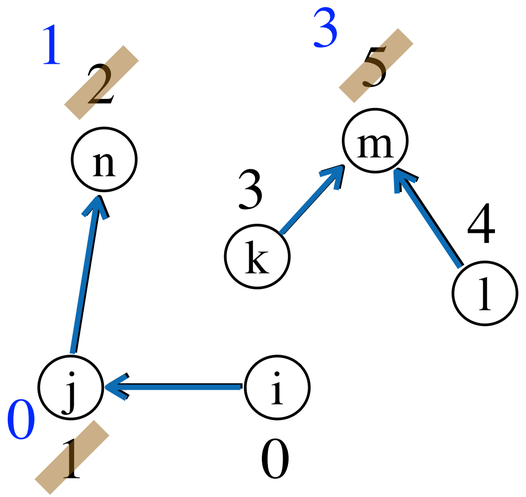
second propagation



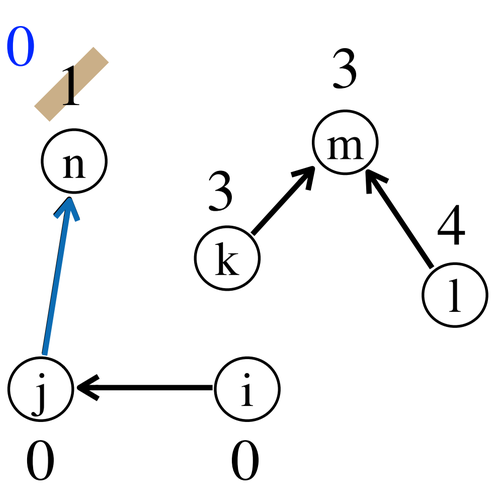
add edge

WCC by propagating labels

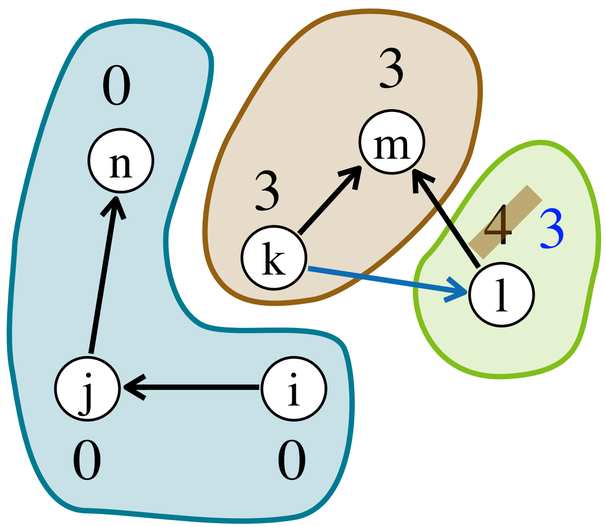
iterative graph algorithm



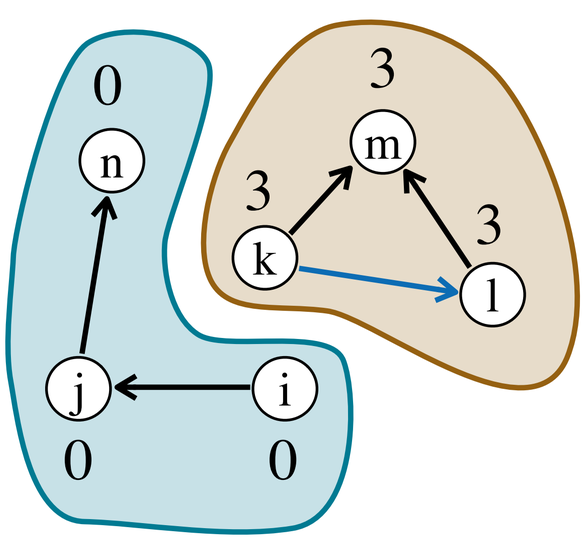
first propagation



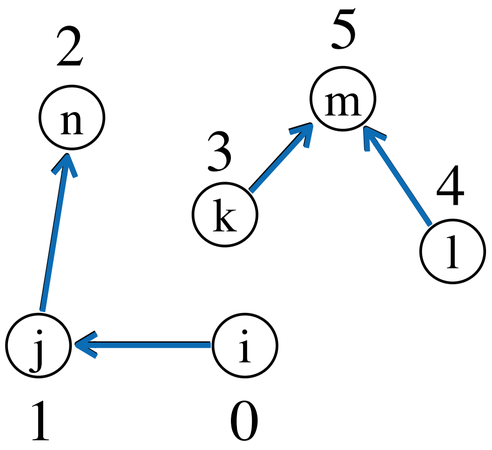
second propagation



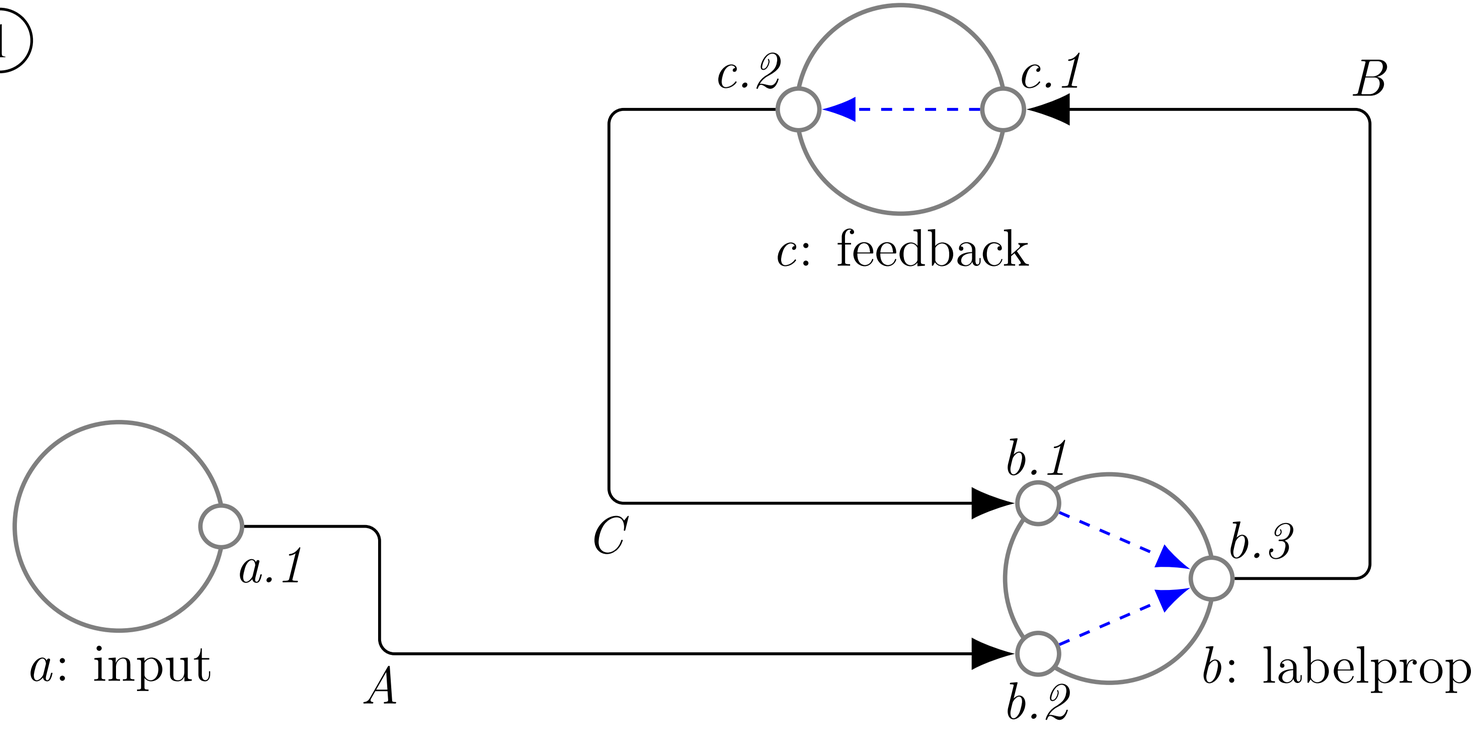
add edge



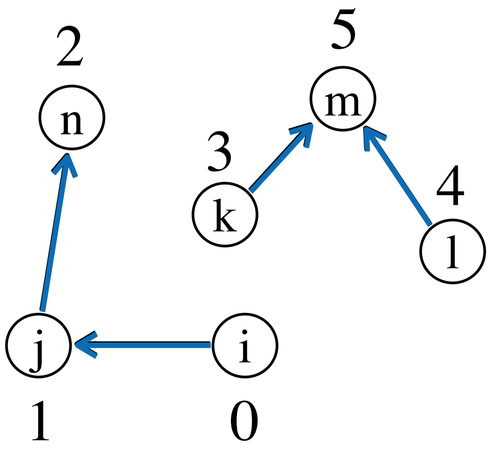
Timely Dataflow supports cyclical dataflows



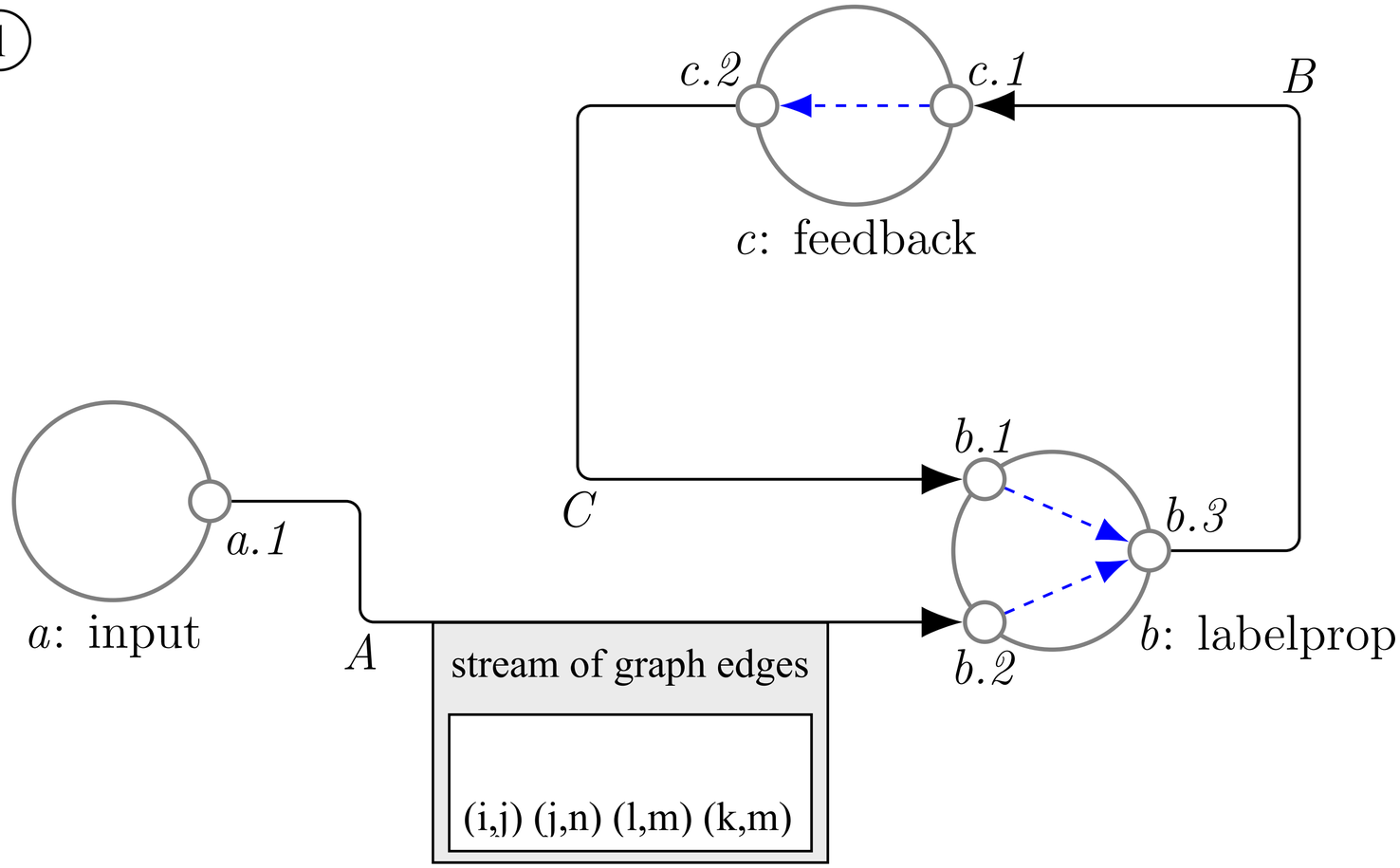
WCC by propagating labels



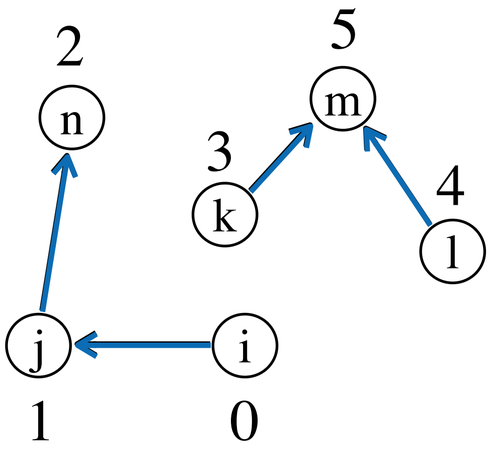
Timely Dataflow supports cyclical dataflows



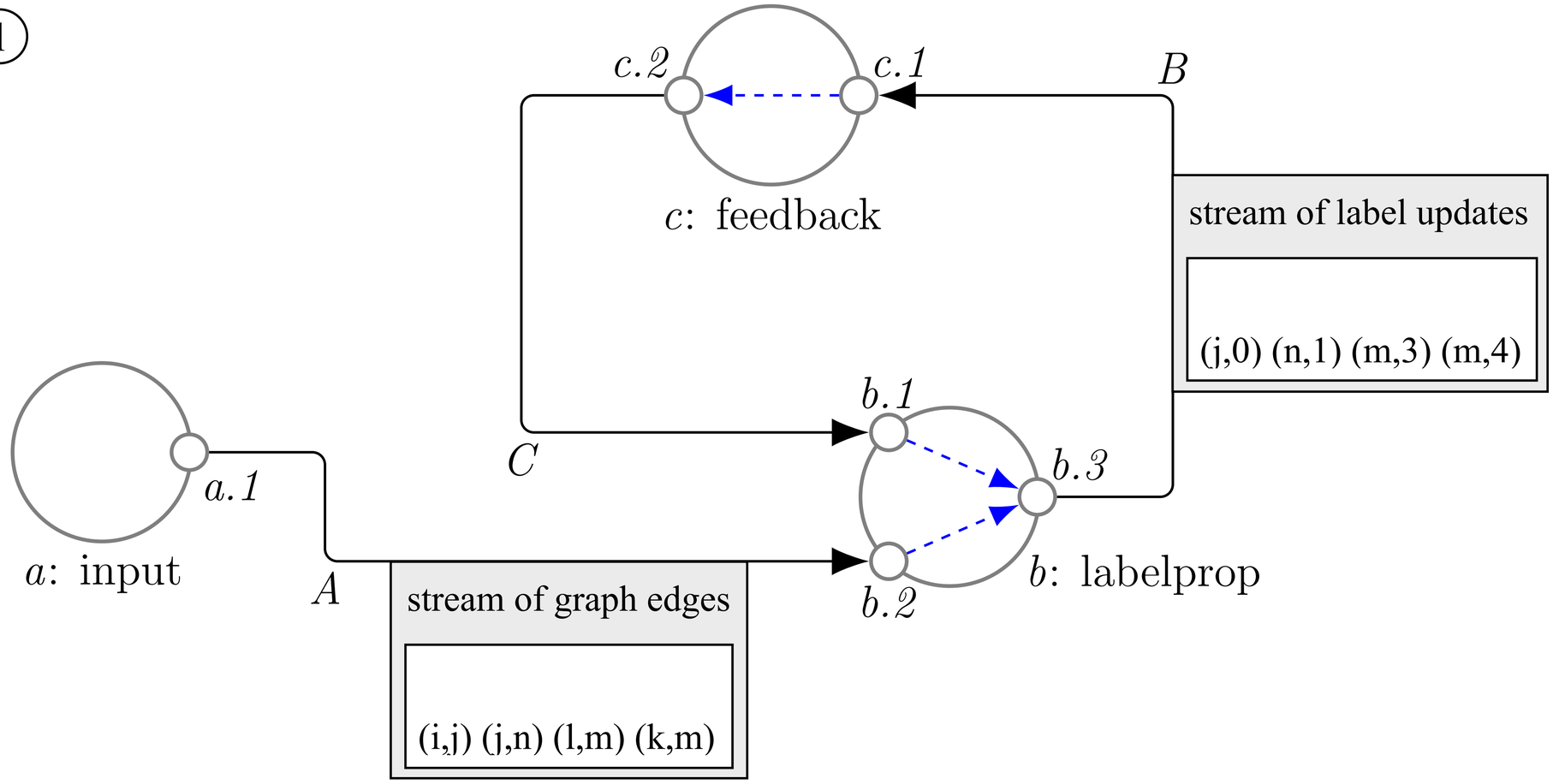
WCC by propagating labels



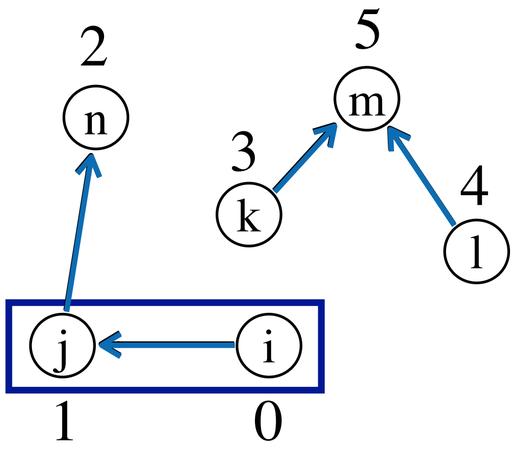
Timely Dataflow supports cyclical dataflows



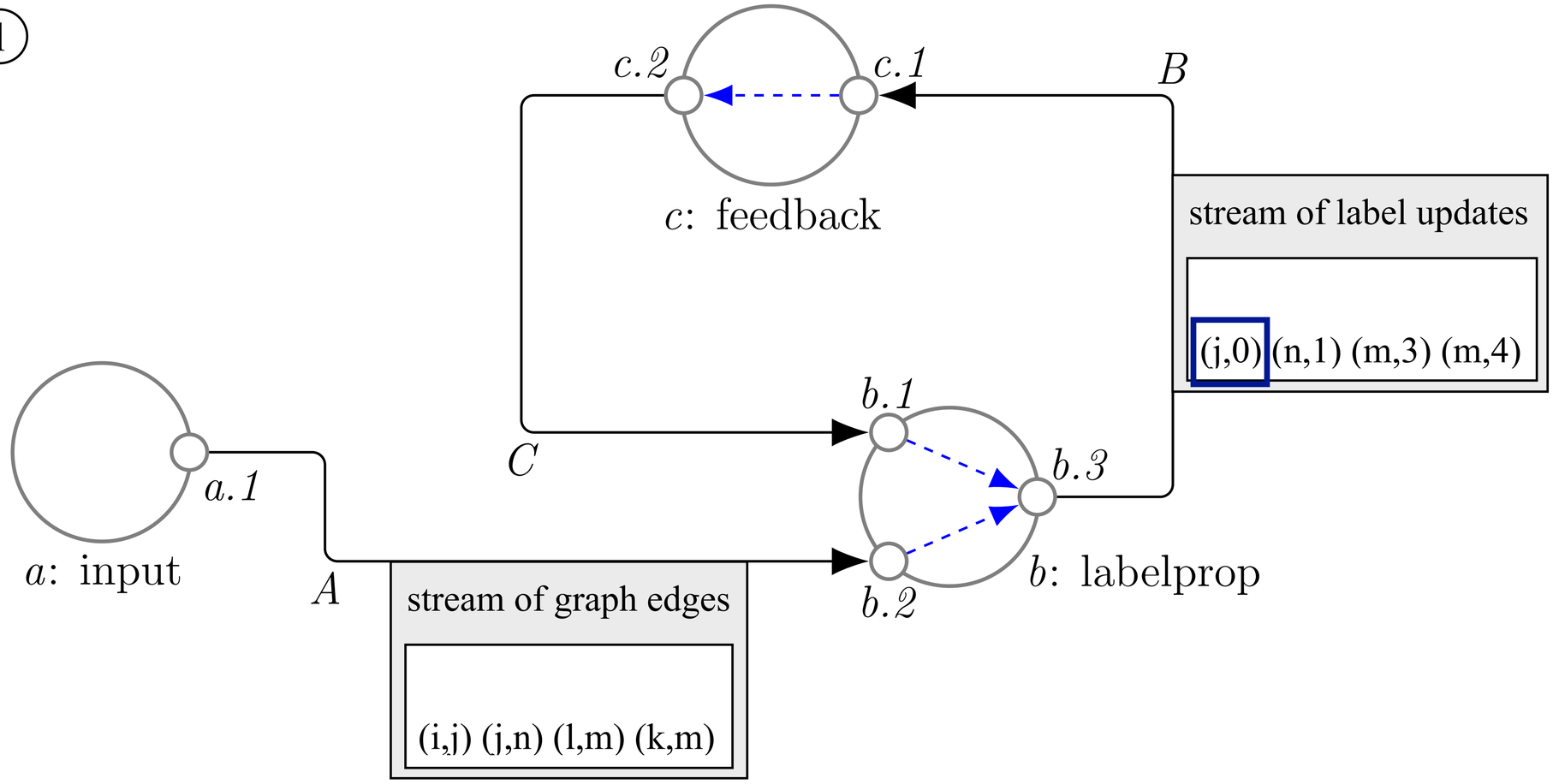
WCC by propagating labels



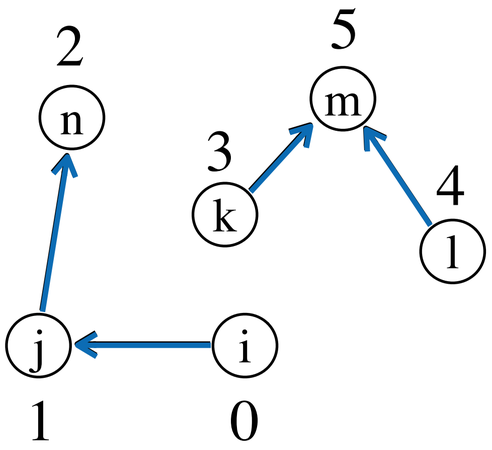
Timely Dataflow supports cyclical dataflows



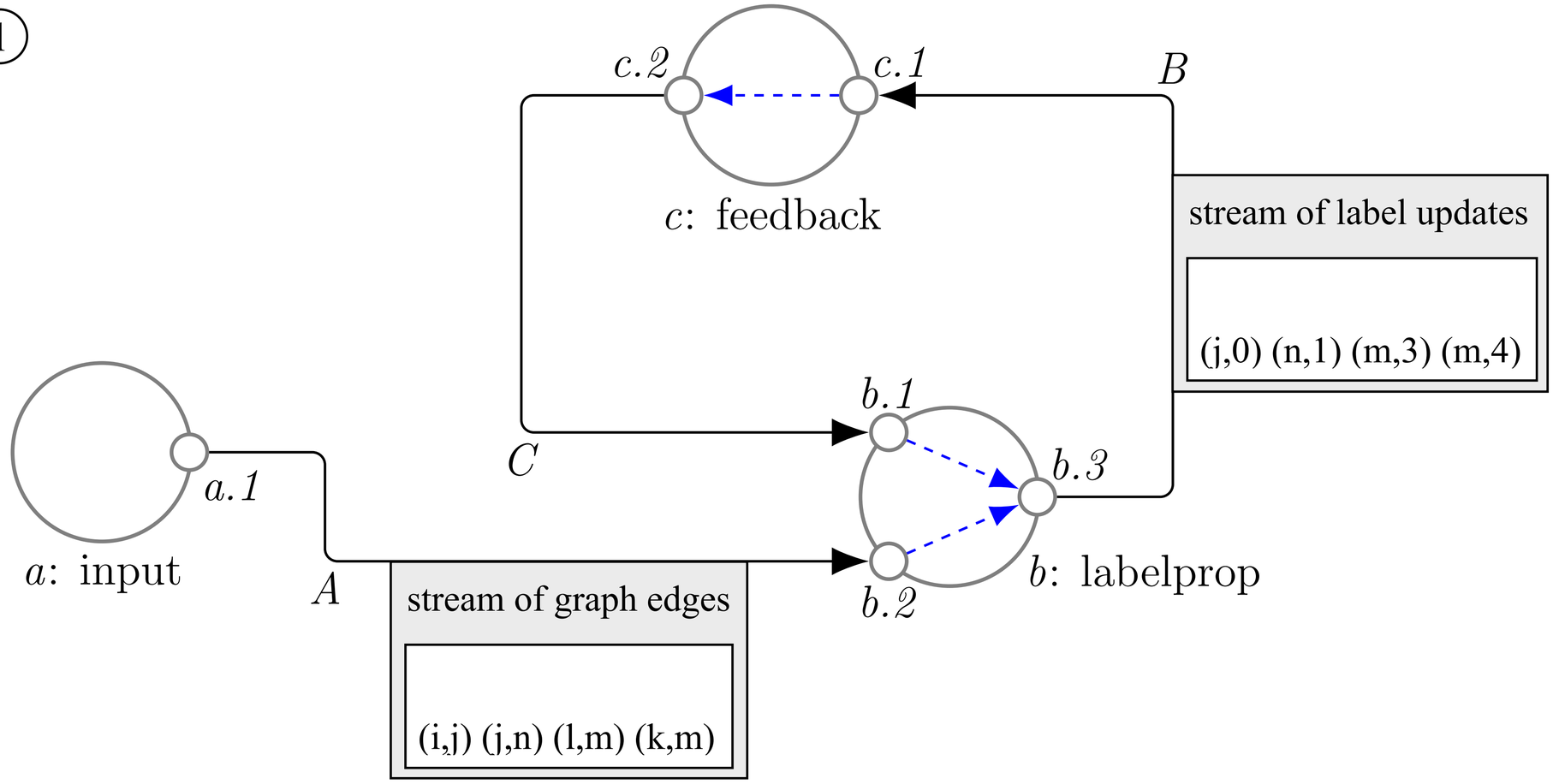
WCC by propagating labels



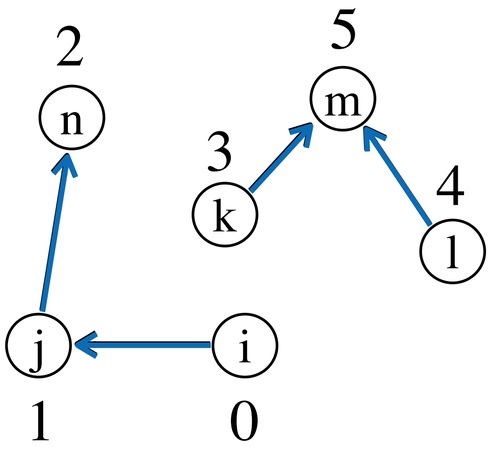
Timely Dataflow supports cyclical dataflows



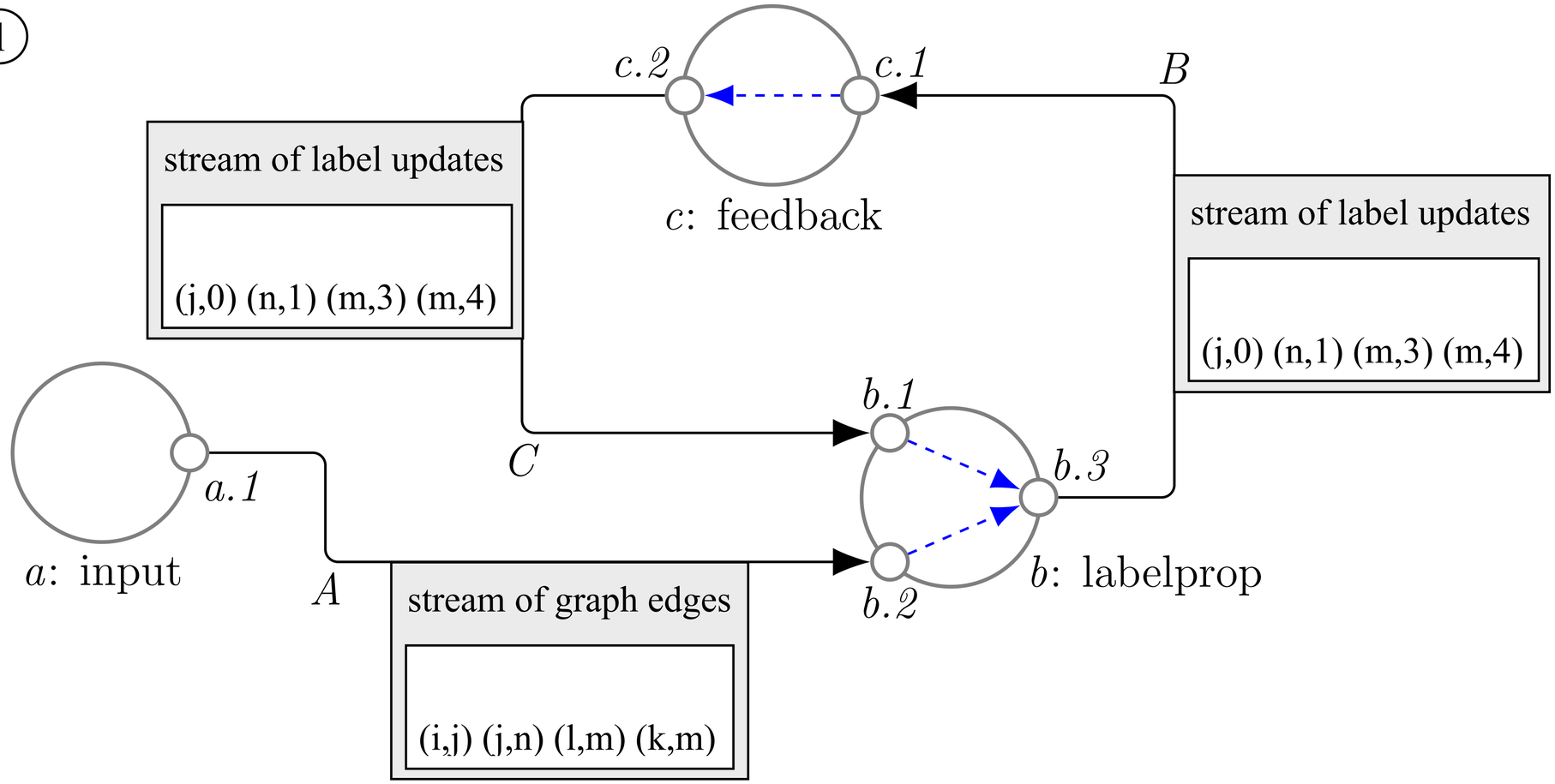
WCC by propagating labels



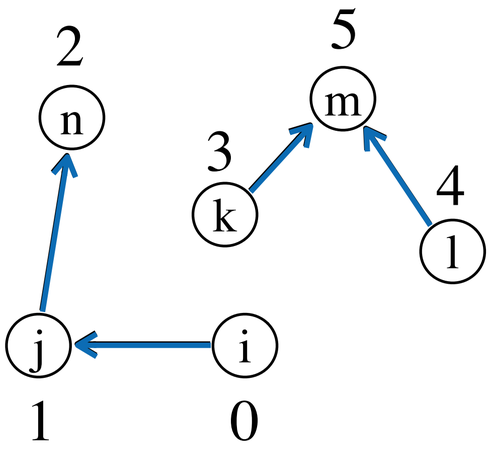
Timely Dataflow supports cyclical dataflows



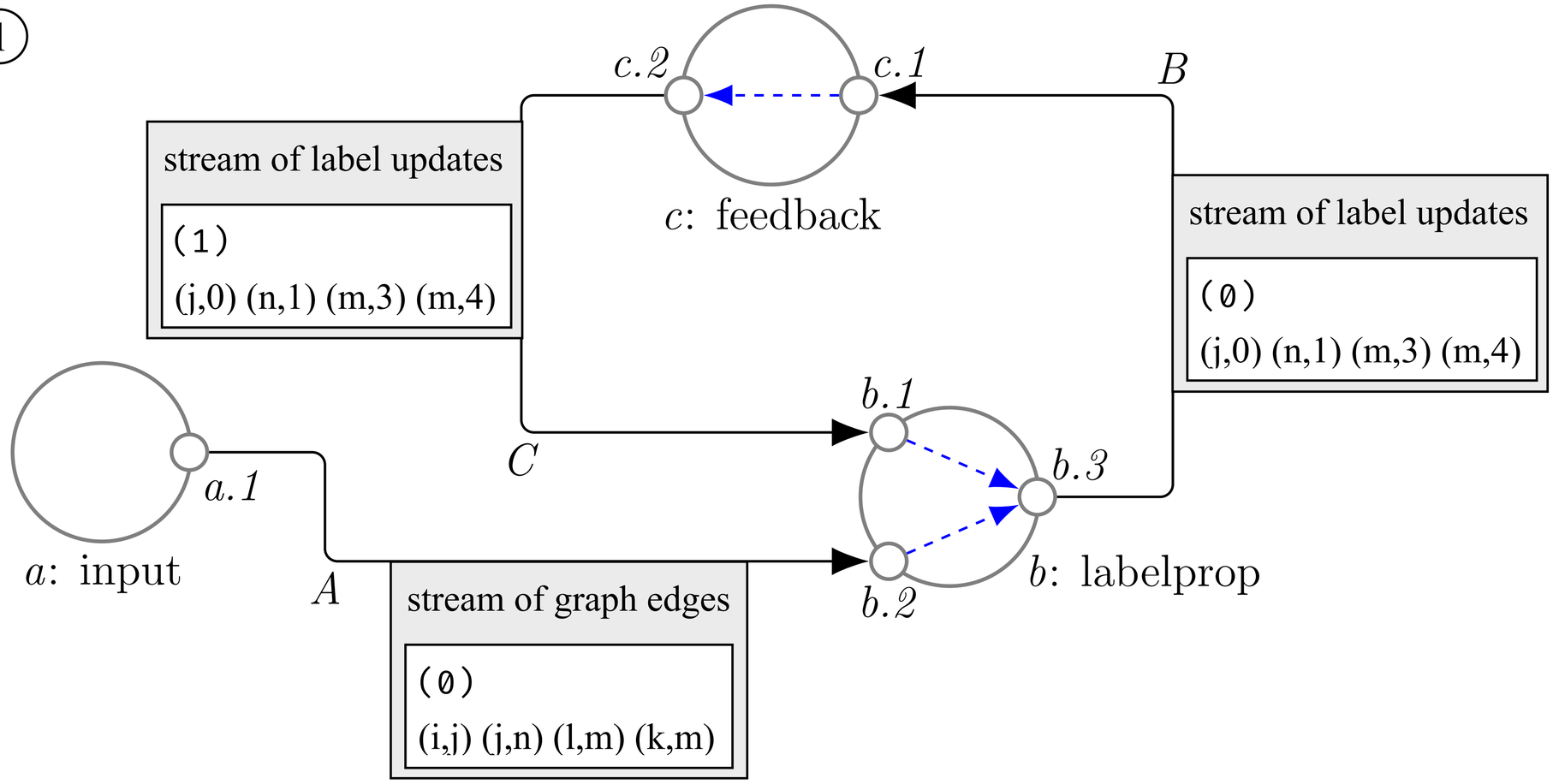
WCC by propagating labels



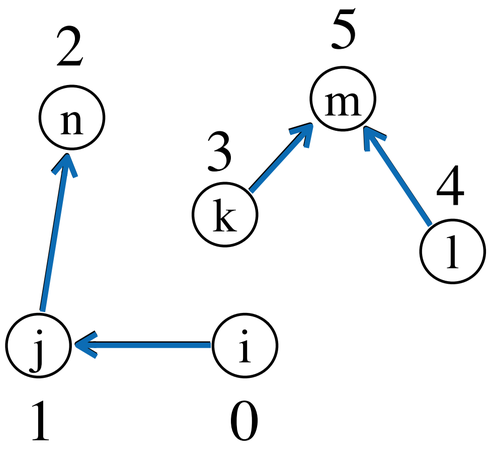
Timely Dataflow supports cyclical dataflows



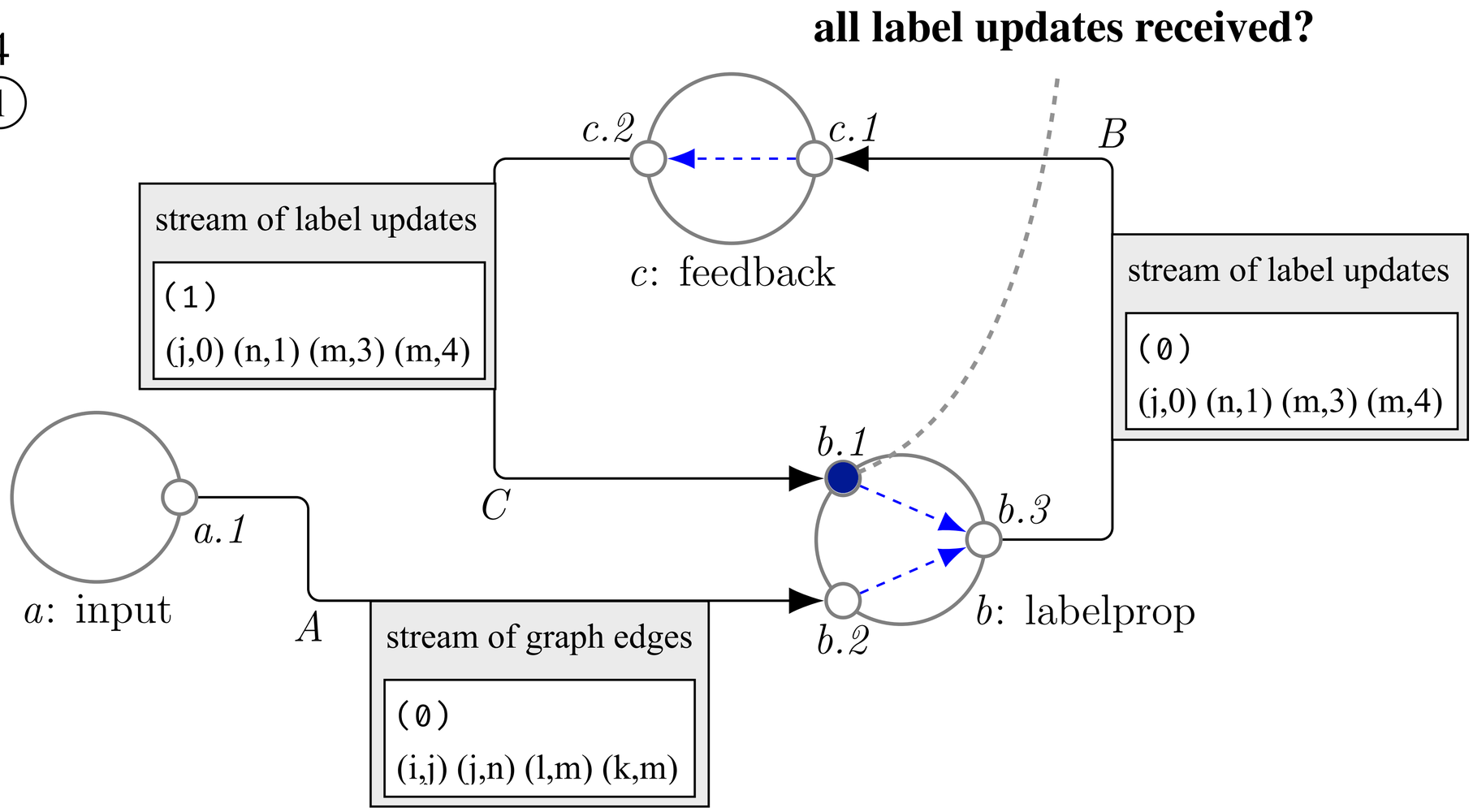
WCC by propagating labels



Timely Dataflow supports cyclical dataflows



WCC by propagating labels

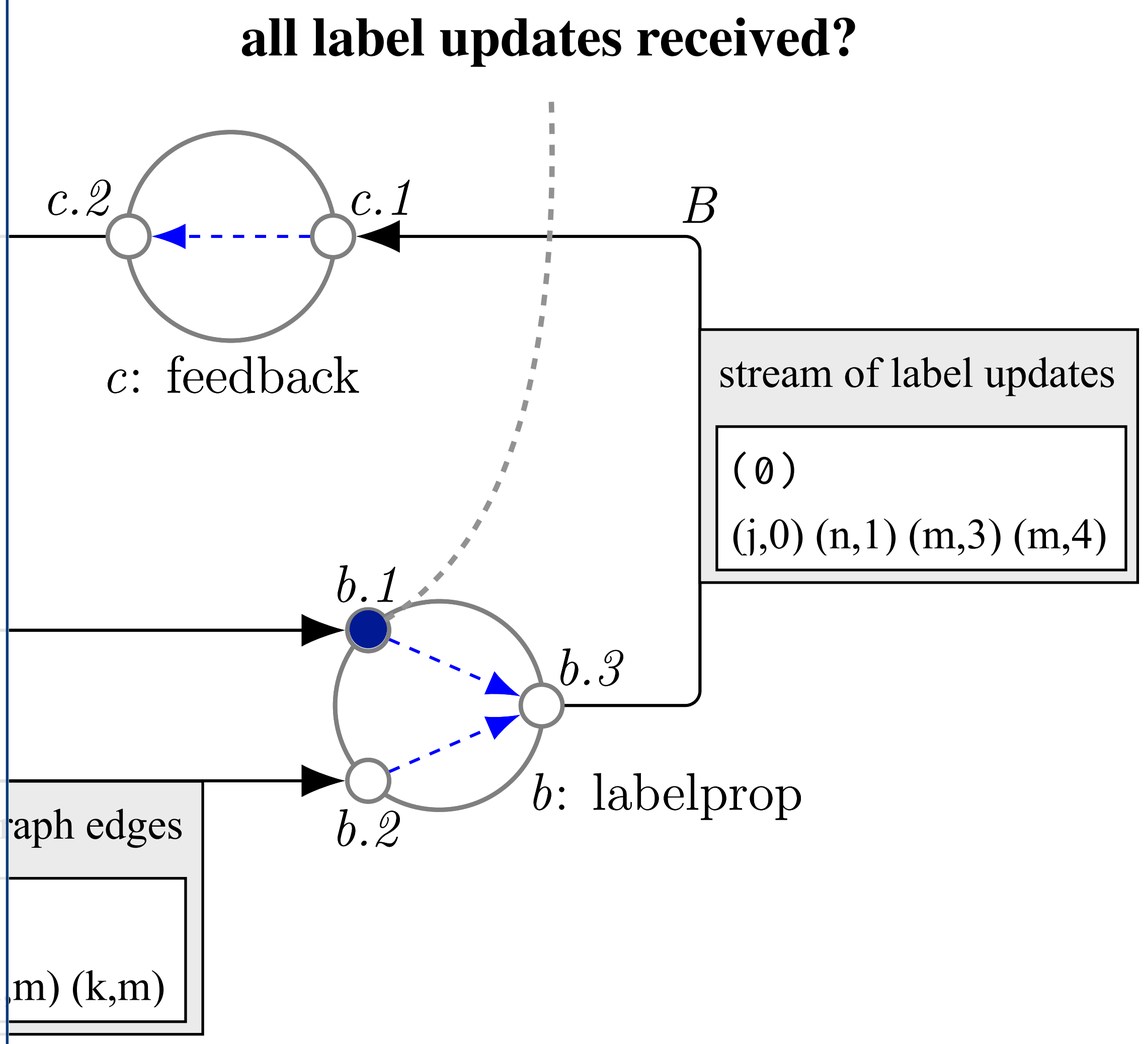


Progress Tracking

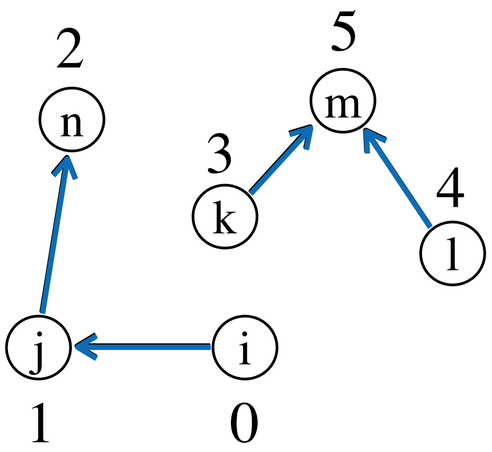
Core coordination component of Timely Dataflow

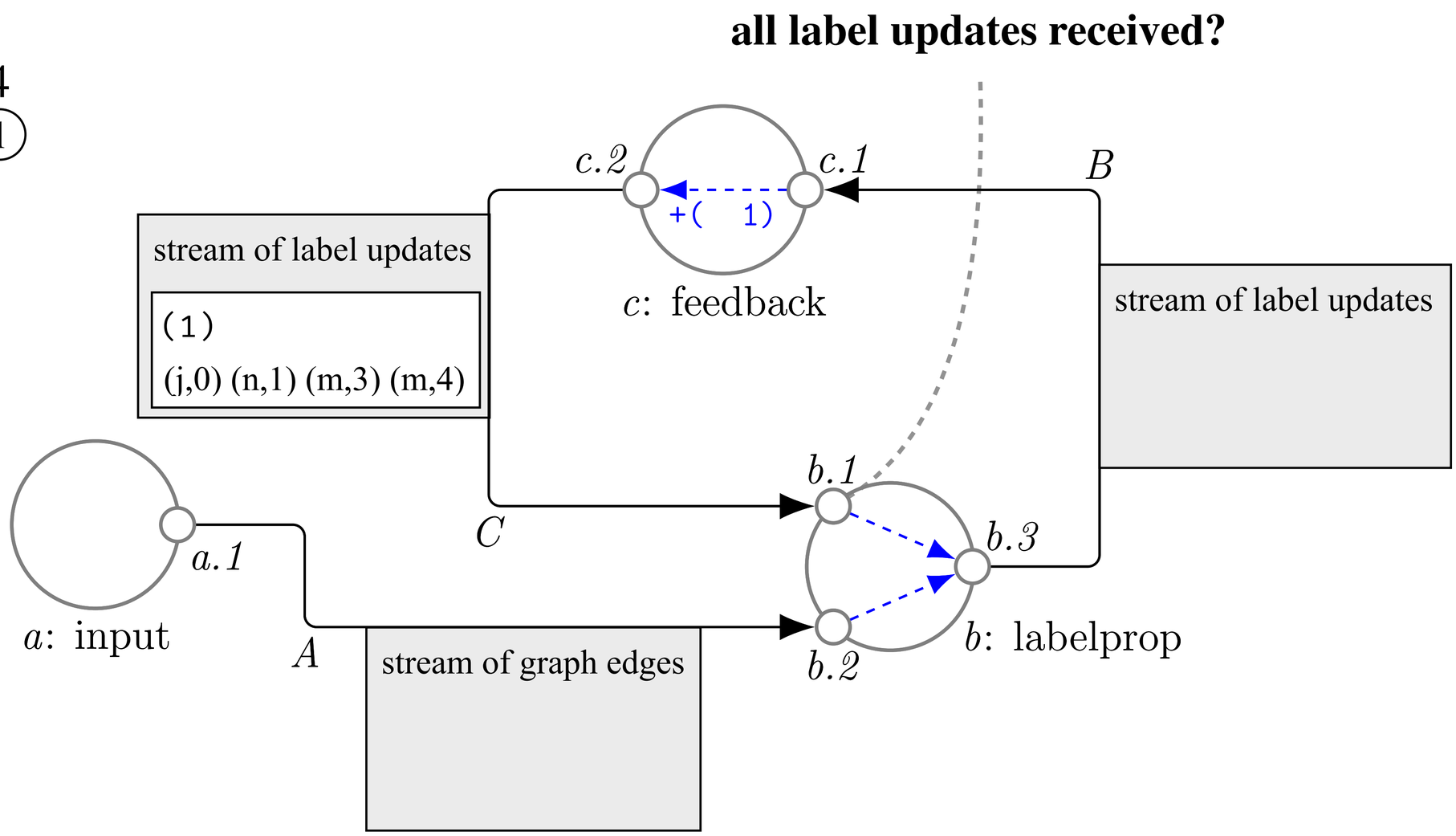
Tracks outstanding work in the system

Informs operators when all data with a certain timestamp has been delivered

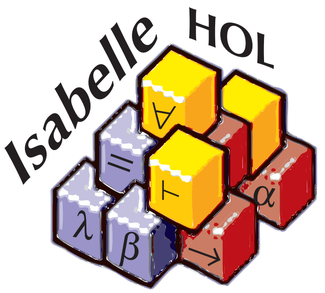


What can go wrong

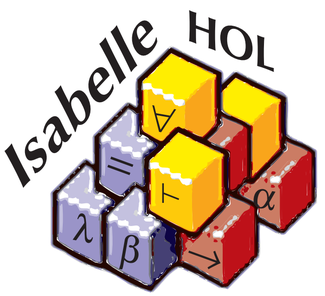




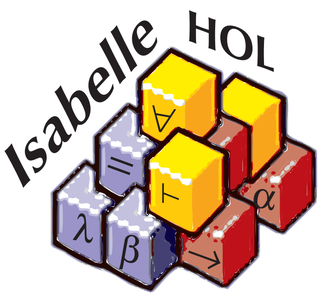
In this work



In this work

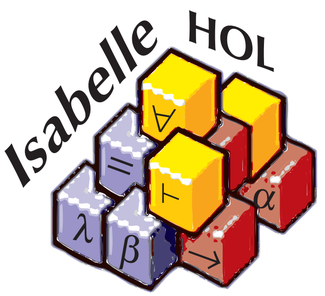


extend the existing model (Abadi et al.) of the distributed component



In this work

extend the existing model (Abadi et al.) of the distributed component model, and verify, the worker-local propagation component



In this work

extend the existing model (Abadi et al.) of the distributed component

model, and verify, the worker-local propagation component

verify safety of the full Progress Tracker

Progress tracking concepts (1)

Progress tracking concepts (1)

Pointstamp (l, t) a datum at rest at an operator, or in motion on a channel

Progress tracking concepts (1)

Pointstamp (l, t) a datum at rest at an operator, or in motion on a channel

Progress tracking concepts (1)

Pointstamp (l, t) a datum at rest at an operator, or in motion on a channel

Timestamp semantic (causal) grouping of the data
tupes of positive integers, or any type with $\preccurlyeq$

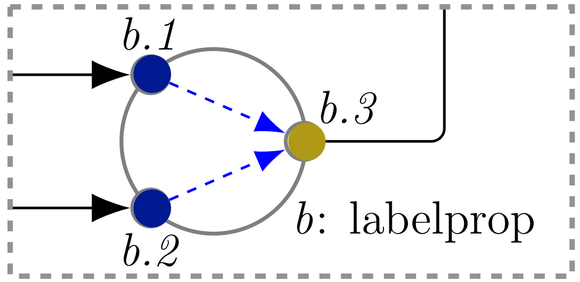
Location

Progress tracking concepts (1)

Pointstamp (l, t) a datum at rest at an operator, or in motion on a channel

Timestamp semantic (causal) grouping of the data
tupes of positive integers, or any type with $\preccurlyeq$

Location operator input or output port

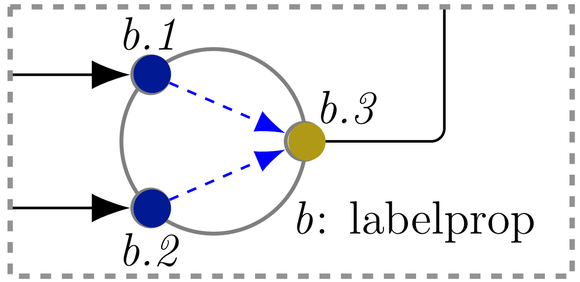


Progress tracking concepts (1)

Pointstamp (l, t) a datum at rest at an operator, or in motion on a channel

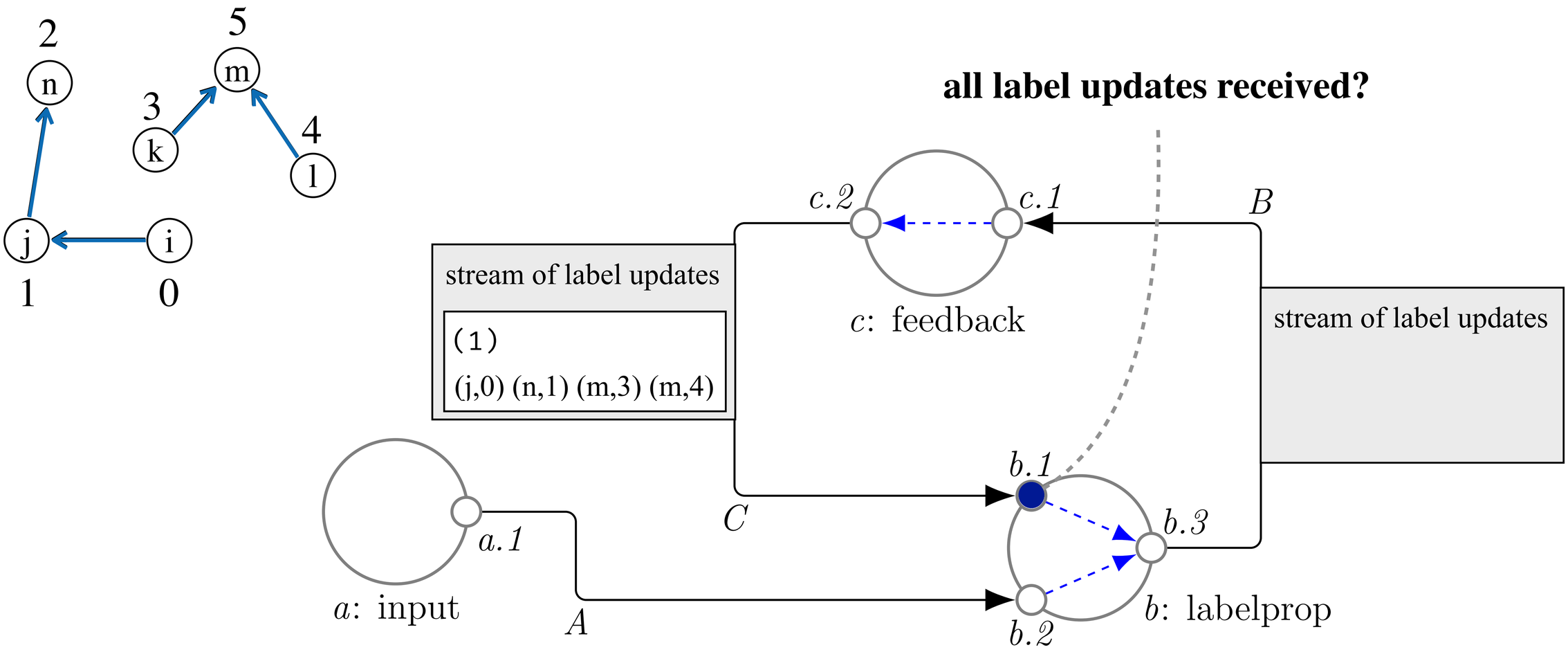
Timestamp semantic (causal) grouping of the data
tupes of positive integers, or any type with $\preccurlyeq$

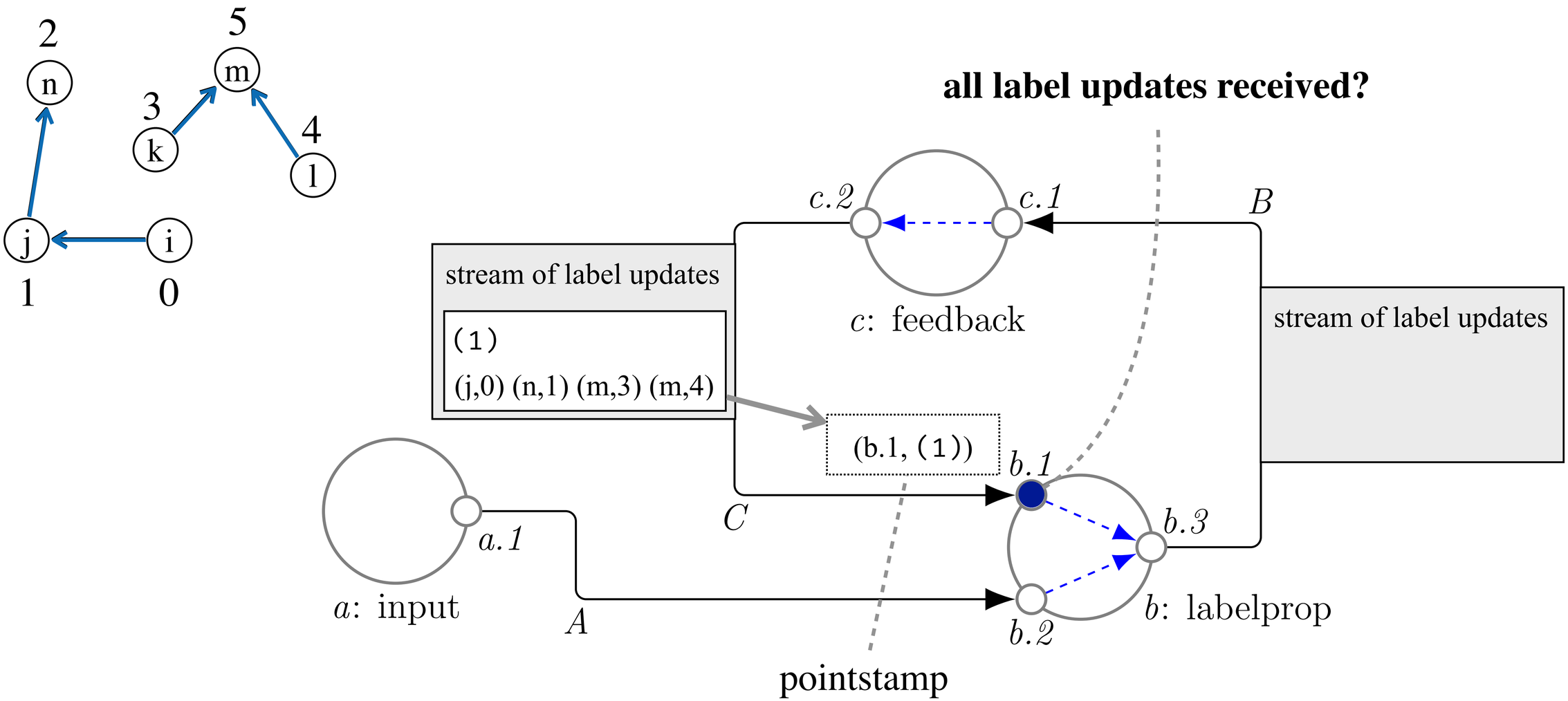
Location operator input or output port

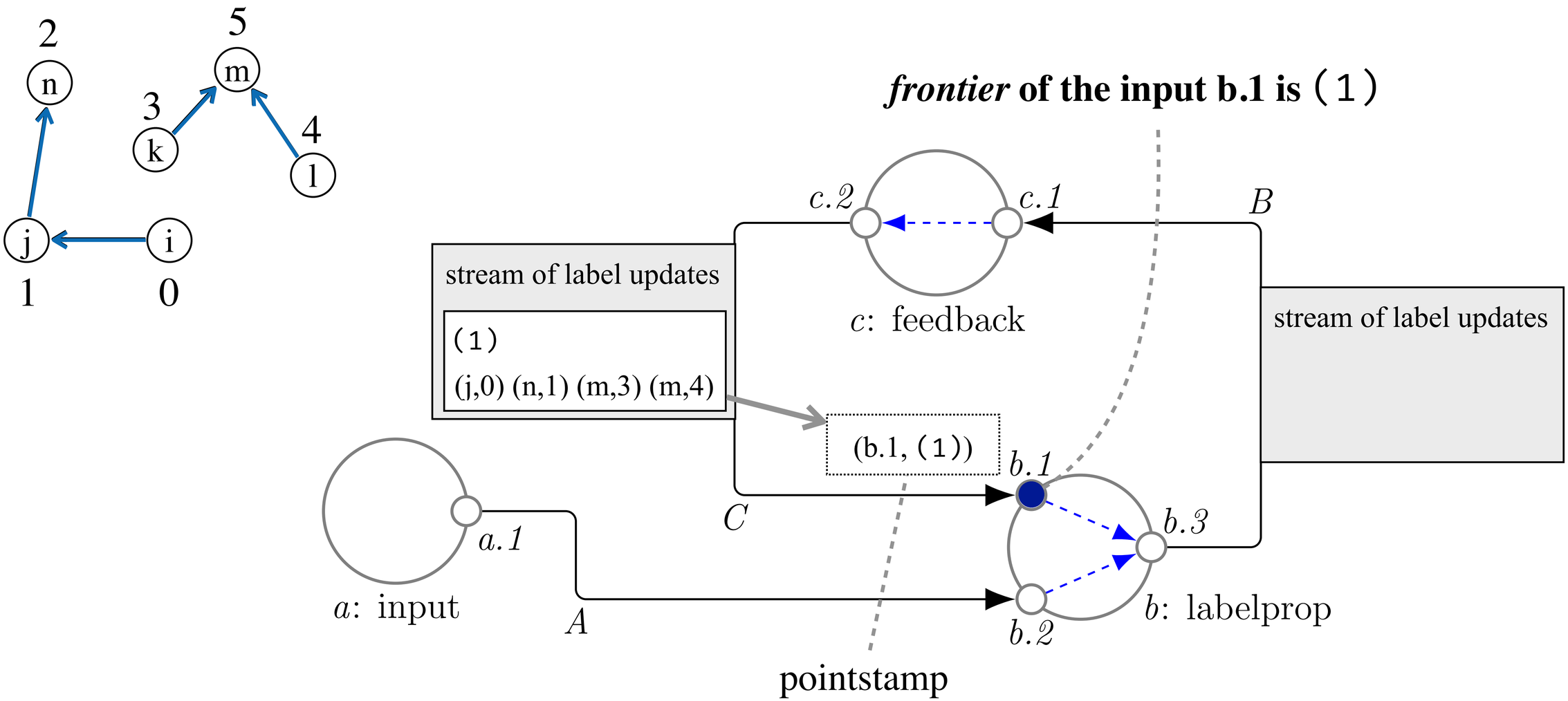


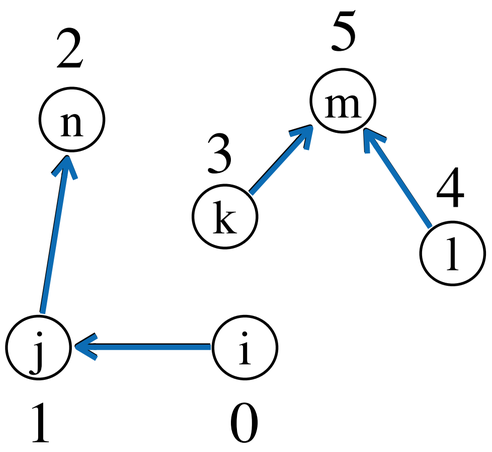
Progress Tracking tracks the pointstamps and summarizes them as a

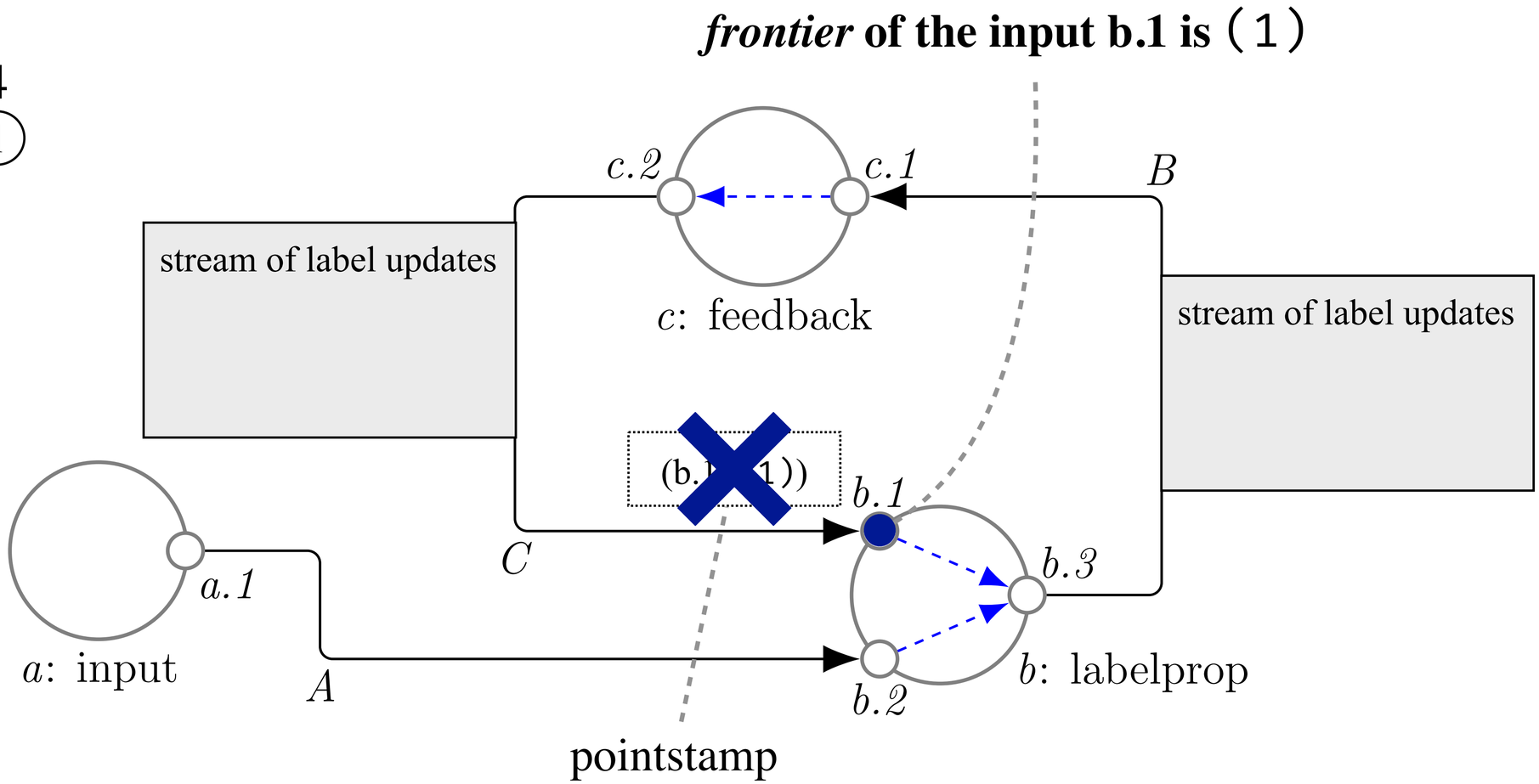
Frontier represented as antichain F
the operator may receive any timestamp t for which $\exists t' \in F.\ t' \preccurlyeq t$

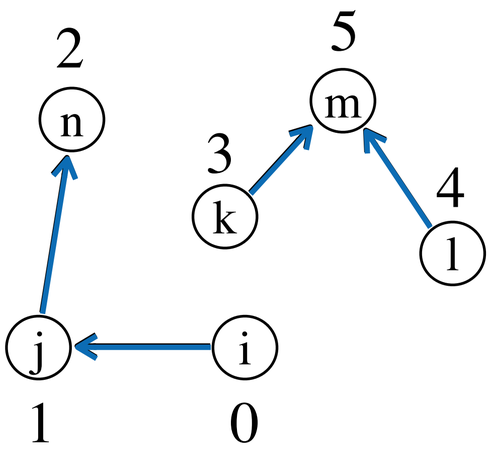

2
n
5
m
3
k
4
l
j
i
1
0
all label updates received?
c.2
c.1
B
stream of label updates
(1)
(j,0) (n,1) (m,3) (m,4)
c: feedback
stream of label updates
b.1
b.3
C
a.1
a: input
A
b.2
b: labelprop


2
n
5
m
3
k
4
l
j
i
1
0
all label updates received?
c.2
c.1
B
stream of label updates
(1)
(j,0) (n,1) (m,3) (m,4)
c: feedback
stream of label updates
(b.1, (1))
b.1
C
b.3
a.1
a: input
A
b.2
b: labelprop
pointstamp


2
n
5
m
3
k
4
l
j
i
1
0
frontier of the input b.1 is (1)
c.2
c.1
B
stream of label updates
(1)
(j,0) (n,1) (m,3) (m,4)
c: feedback
stream of label updates
(b.1, (1))
b.1
C
a.1
b.3
a: input
A
b.2
b: labelprop
pointstamp


2
n
5
m
3
k
4
l
j
i
1
0


frontier of the input b.1 is (1)
c.2
c.1
B
stream of label updates
c: feedback
stream of label updates
(b.1 1))
b.1
C
a.1
b.3
a: input
A
b.2
b: labelprop
pointstamp



***frontier* of the input b.1 is (2)**

stream of label updates

c.2

c.1

B

c: feedback

stream of label updates

(b.1 1))

b.1

C

a.1

b.3

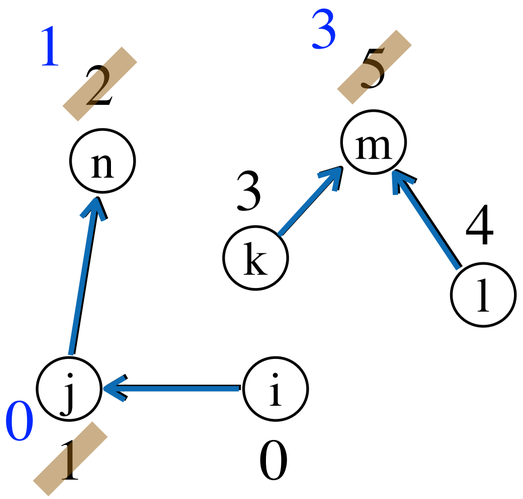
a: input

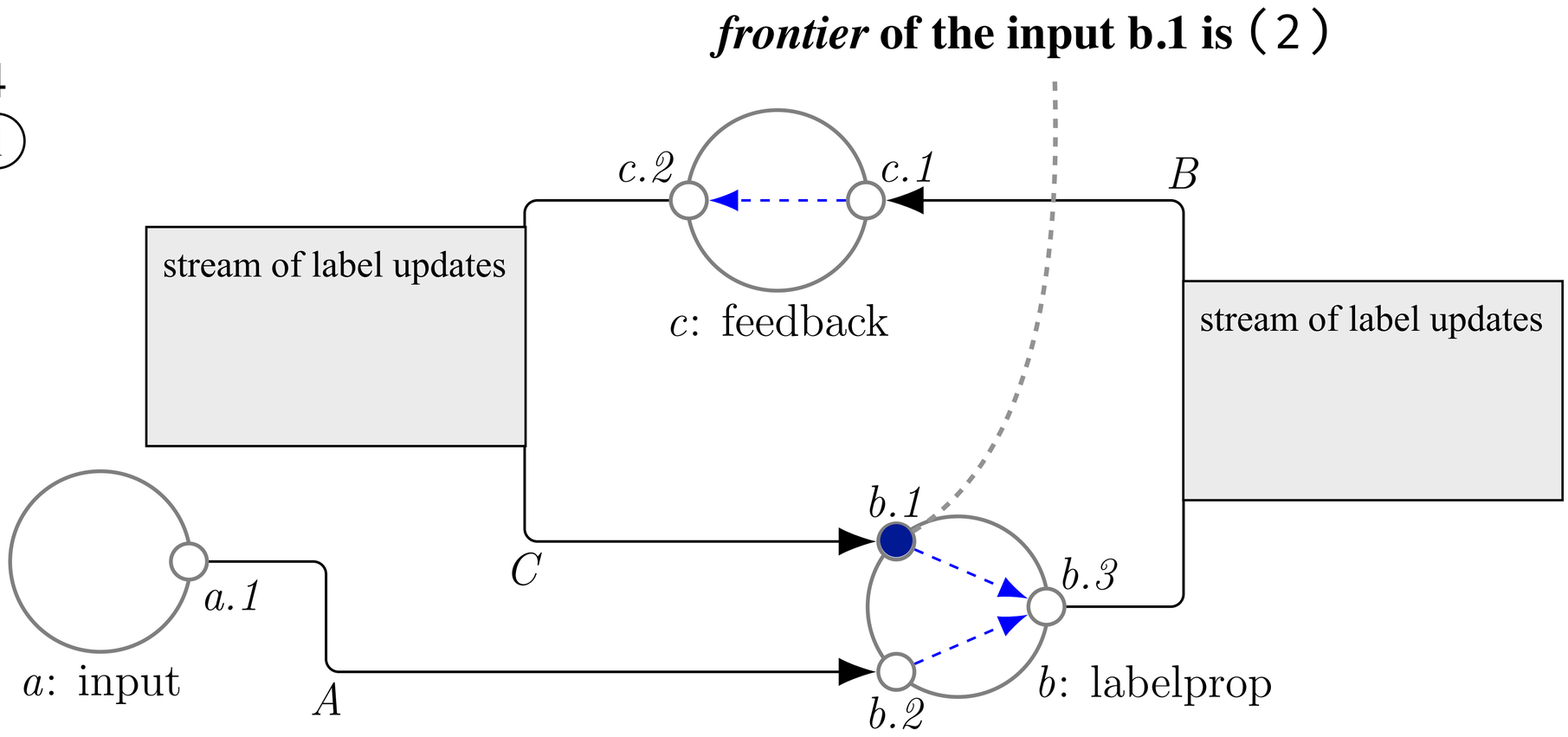
A

b.2

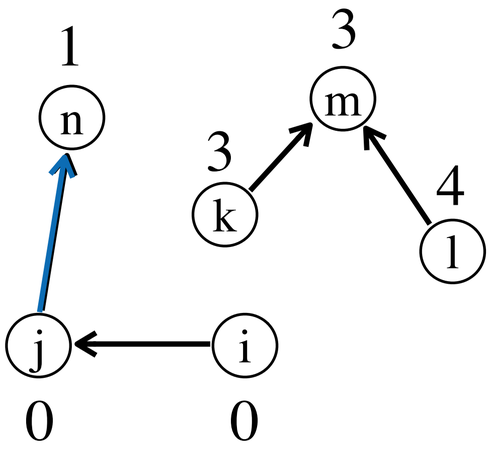
b: labelprop

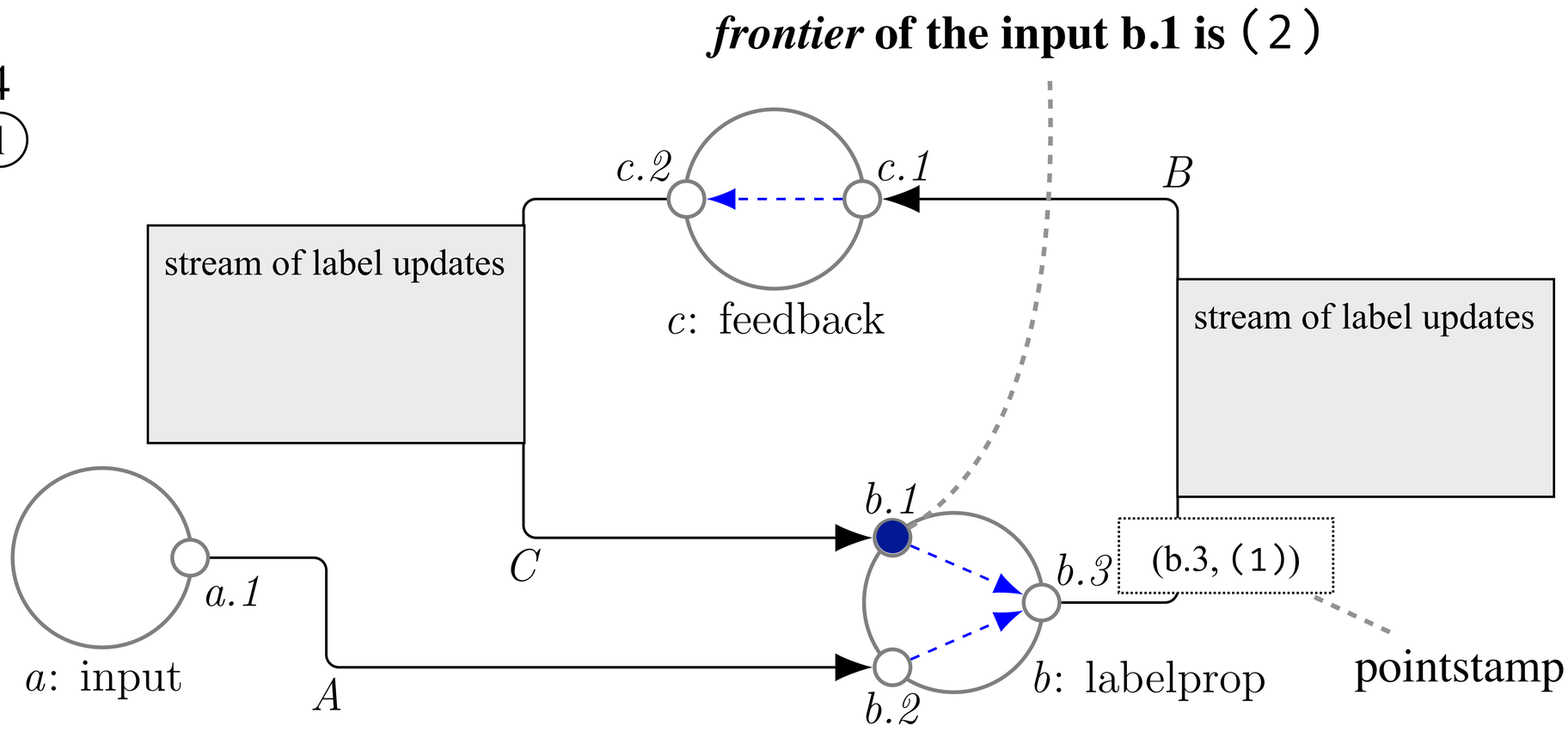
pointstamp


1
2
3
5
n
m
3
k
4
l
0
j
i
1
0

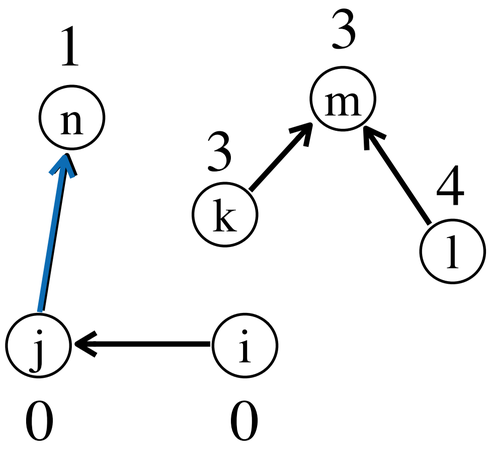

frontier of the input b.1 is (2)
c.2
c.1
B
stream of label updates
c: feedback
stream of label updates
b.1
C
b.3
a.1
a: input
A
b.2
b: labelprop

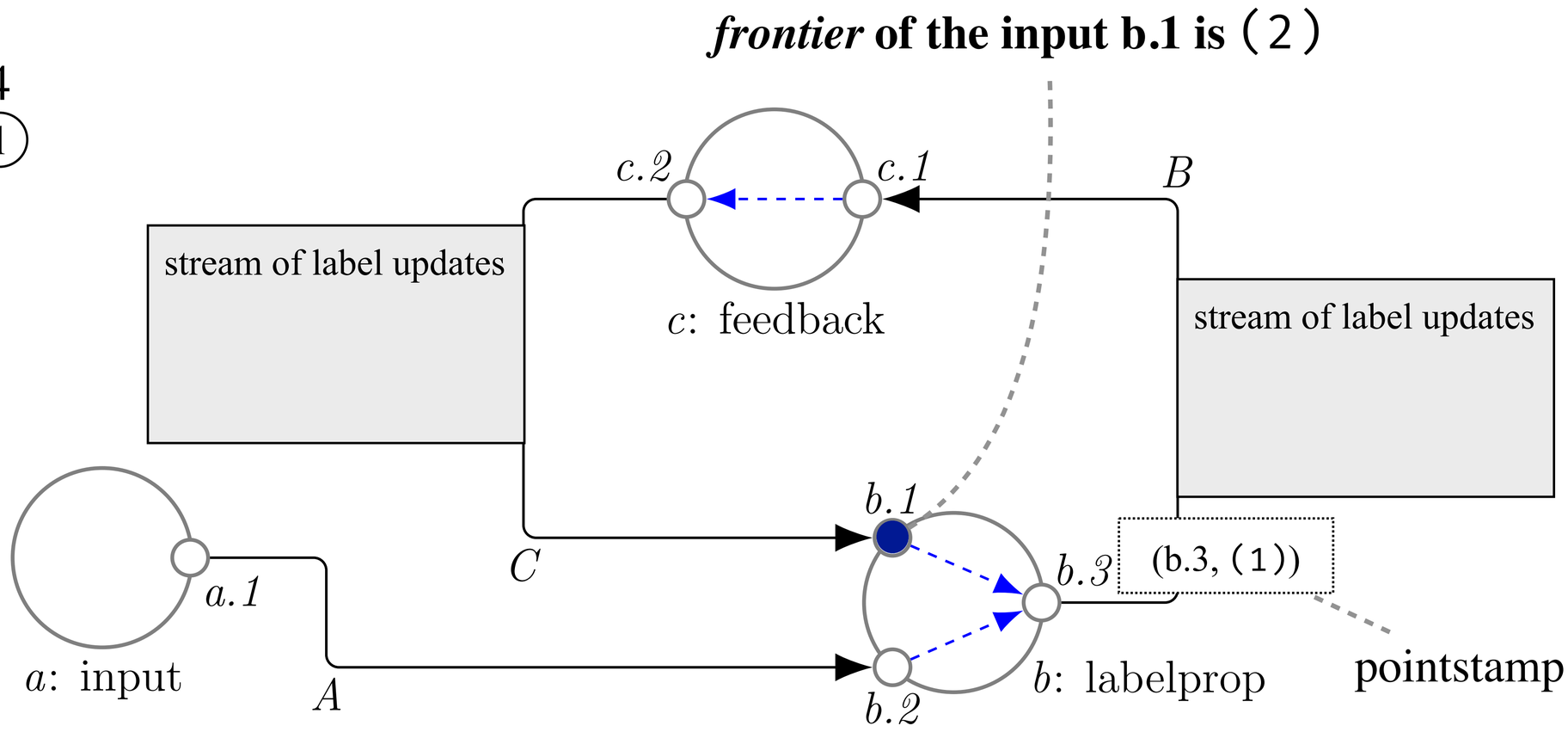
Progress tracking concepts (2)





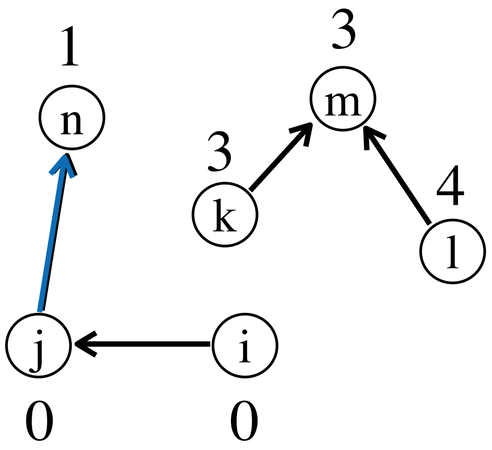
Progress tracking concepts (2)

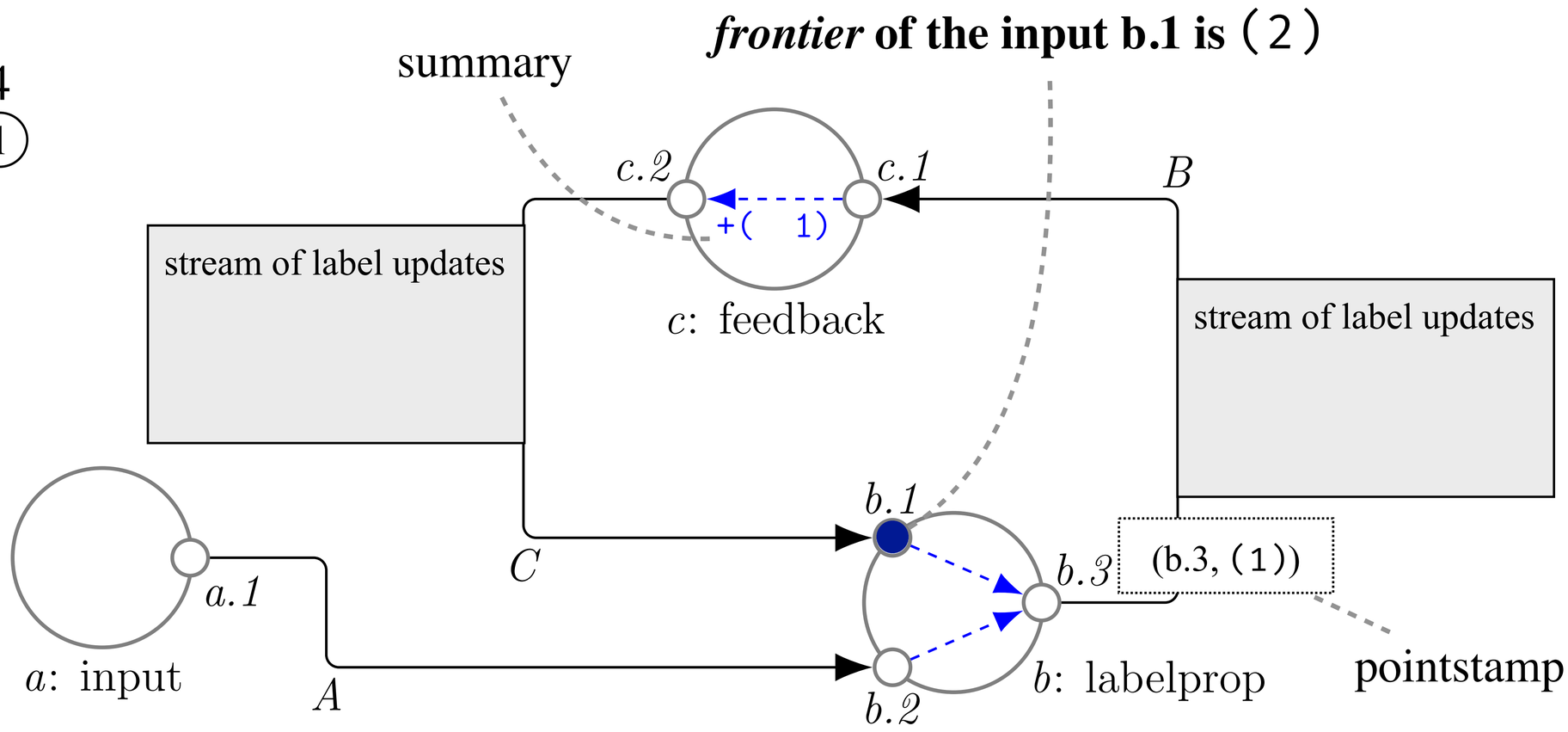




Summary minimal increment to timestamps applied to data processed by the operator

Progress tracking concepts (2)



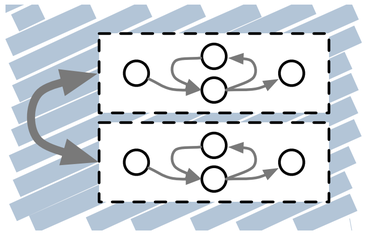


Summary minimal increment to timestamps applied to data processed by the operator

Progress tracking

computes frontiers in two steps

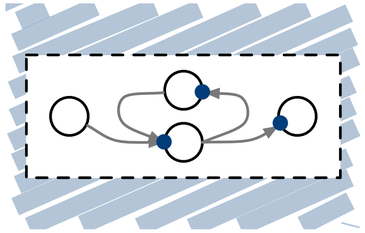
1



distributed component **exchanges** pointstamp changes

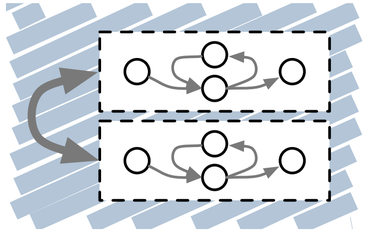
↳ approximate, conservative view of all pointstamps

2



worker-local component **propagates** changes on the dataflow graph

↳ update frontiers at operator input ports



Abadi et al.'s exchange protocol

Formal Analysis of a Distributed Algorithm for Tracking Progress

Martín Abadi[1,2], Frank McSherry[1], Derek G. Murray[1], and Thomas L. Rodeheffer[1]

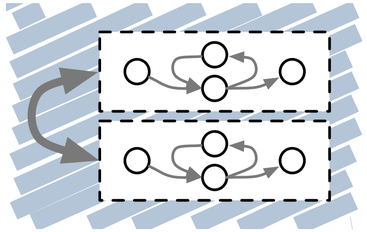
[1] Microsoft Research Silicon Valley
[2] University of California, Santa Cruz

Abstract. Tracking the progress of computations can be both important and delicate in distributed systems. In a recent distributed algorithm for this purpose, each processor maintains a delayed view of the pending work, which is represented in terms of points in virtual time. This paper presents a formal specification of that algorithm in the temporal logic TLA, and describes a mechanically verified correctness proof of its main properties.

Formalized in TLA+PS before

We ported the formalization to Isabelle

One person-week, *Sketch & Hammer*



Abadi et al.'s exchange protocol

Formal Analysis of a Distributed Algorithm for Tracking Progress

Martín Abadi[1,2], Frank McSherry[1], Derek G. Murray[1], and Thomas L. Rodeheffer[1]

[1] Microsoft Research Silicon Valley
[2] University of California, Santa Cruz

Abstract. Tracking the progress of computations can be both important and delicate in distributed systems. In a recent distributed algorithm for this purpose, each processor maintains a delayed view of the pending work, which is represented in terms of points in virtual time. This paper presents a formal specification of that algorithm in the temporal logic TLA, and describes a mechanically verified correctness proof of its main properties.

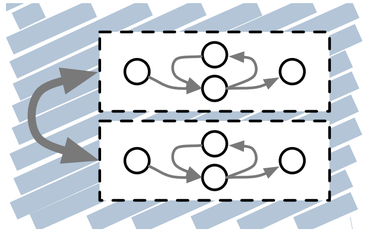
Formalized in TLA+PS before

We ported the formalization to Isabelle

One person-week, *Sketch & Hammer*

Discovered problems:

1. Transitions based on global state
2. Too strong restrictions on allowed transitions



Abadi et al.'s exchange protocol

Formal Analysis of a Distributed Algorithm for Tracking Progress

Martín Abadi[1,2], Frank McSherry[1], Derek G. Murray[1], and Thomas L. Rodeheffer[1]

[1] Microsoft Research Silicon Valley
[2] University of California, Santa Cruz

Abstract. Tracking the progress of computations can be both important and delicate in distributed systems. In a recent distributed algorithm for this purpose, each processor maintains a delayed view of the pending work, which is represented in terms of points in virtual time. This paper presents a formal specification of that algorithm in the temporal logic TLA, and describes a mechanically verified correctness proof of its main properties.

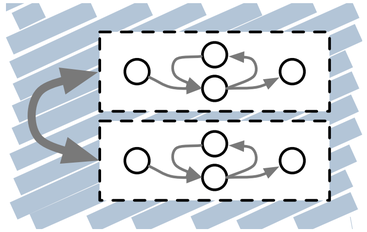
Formalized in TLA+PS before

We ported the formalization to Isabelle

One person-week, *Sketch & Hammer*

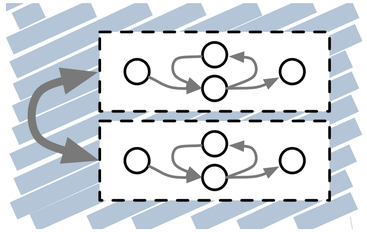
Discovered problems:

1. Transitions based on global state
2. Too strong restrictions on allowed transitions



Problem 1: Accessing global state

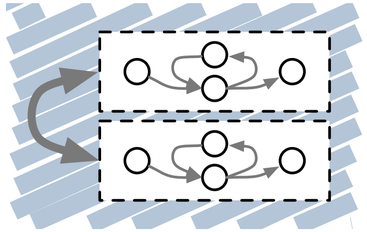
$$\begin{aligned}&\textbf{record}\ ('w :: \mathit{finite}, 'p :: \mathit{order})\ \mathit{conf} =\\ &\quad \mathsf{rec} :: 'p\ \mathit{zmset}\\ &\quad \mathsf{msg} :: 'w \Rightarrow 'w \Rightarrow 'p\ \mathit{zmset}\ \mathit{list}\\ &\quad \mathsf{temp} :: 'w \Rightarrow 'p\ \mathit{zmset}\\ &\quad \mathsf{glob} :: 'w \Rightarrow 'p\ \mathit{zmset}\end{aligned}$$



Problem 1: Accessing global state

Finite set of workers

$$\textbf{record}\ ('w :: \mathit{finite}, 'p :: \mathit{order})\ \mathit{conf} =$$
$$\quad \mathsf{rec} :: 'p\ \mathit{zmset}$$
$$\quad \mathsf{msg} :: 'w \Rightarrow 'w \Rightarrow 'p\ \mathit{zmset}\ \mathit{list}$$
$$\quad \mathsf{temp} :: 'w \Rightarrow 'p\ \mathit{zmset}$$
$$\quad \mathsf{glob} :: 'w \Rightarrow 'p\ \mathit{zmset}$$

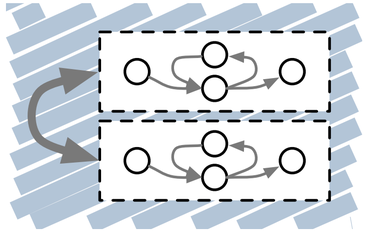


Problem 1: Accessing global state

Finite set of workers

Partially-ordered pointstamps

$$\textbf{record}\ ('w :: \mathit{finite}, 'p :: \mathit{order})\ \mathit{conf} =$$
$$\quad \mathsf{rec} :: 'p\ \mathit{zmset}$$
$$\quad \mathsf{msg} :: 'w \Rightarrow 'w \Rightarrow 'p\ \mathit{zmset}\ \mathit{list}$$
$$\quad \mathsf{temp} :: 'w \Rightarrow 'p\ \mathit{zmset}$$
$$\quad \mathsf{glob} :: 'w \Rightarrow 'p\ \mathit{zmset}$$



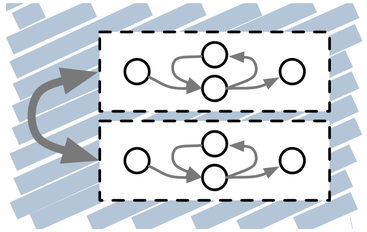
Problem 1: Accessing global state

Finite set of workers

Partially-ordered pointstamps

$$\textbf{record}\ ('w :: \mathit{finite}, 'p :: \mathit{order})\ \mathit{conf} =$$
$$\quad \textsf{rec} :: 'p\ \mathit{zmset}$$
$$\quad \textsf{msg} :: 'w \Rightarrow 'w \Rightarrow 'p\ \mathit{zmset}\ \mathit{list}$$
$$\quad \textsf{temp} :: 'w \Rightarrow 'p\ \mathit{zmset}$$
$$\quad \textsf{glob} :: 'w \Rightarrow 'p\ \mathit{zmset}$$

signed multiset



Problem 1: Accessing global state

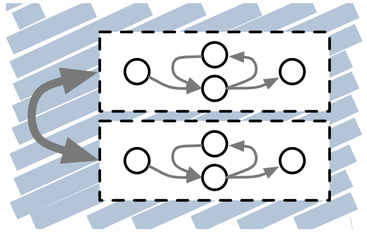
Finite set of workers

Partially-ordered pointstamps

All pointstamps present in the system

signed multiset

$$\textbf{record}\ ('w :: \mathit{finite}, 'p :: \mathit{order})\ \mathit{conf} =$$
$$\textbf{rec} :: 'p\ \mathit{zmset}$$
$$\textbf{msg} :: 'w \Rightarrow 'w \Rightarrow 'p\ \mathit{zmset}\ \mathit{list}$$
$$\textbf{temp} :: 'w \Rightarrow 'p\ \mathit{zmset}$$
$$\textbf{glob} :: 'w \Rightarrow 'p\ \mathit{zmset}$$



Problem 1: Accessing global state

Finite set of workers

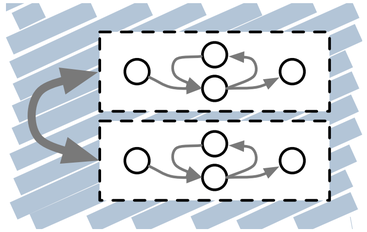
Partially-ordered pointstamps

All pointstamps present in the system

FIFO communication channels

signed multiset

$$\textbf{record}\ ('w :: finite, 'p :: order)\ conf =$$
$$\textsf{rec} :: 'p\ zmset$$
$$\textsf{msg} :: 'w \Rightarrow 'w \Rightarrow 'p\ zmset\ list$$
$$\textsf{temp} :: 'w \Rightarrow 'p\ zmset$$
$$\textsf{glob} :: 'w \Rightarrow 'p\ zmset$$



Problem 1: Accessing global state

Finite set of workers

Partially-ordered pointstamps

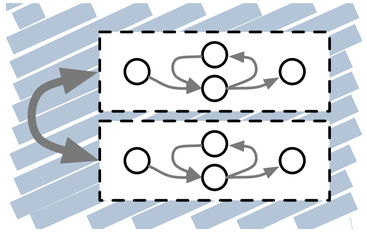
All pointstamps present in the system

FIFO communication channels

$$\textbf{record}\ ('w :: finite, 'p :: order)\ conf =$$
$$\textbf{rec} :: 'p\ zmset$$
$$\textbf{msg} :: 'w \Rightarrow 'w \Rightarrow 'p\ zmset\ list$$
$$\textbf{temp} :: 'w \Rightarrow 'p\ zmset$$
$$\textbf{glob} :: 'w \Rightarrow 'p\ zmset$$

signed multiset

Not-yet-communicated pointstamp updates



Problem 1: Accessing global state

Finite set of workers

Partially-ordered pointstamps

All pointstamps present in the system

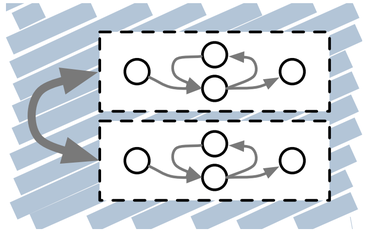
signed multiset

FIFO communication channels

Not-yet-communicated pointstamp updates

Each worker's local approximation of the global state

$$\textbf{record}\ ('w :: \mathit{finite}, 'p :: \mathit{order})\ \mathit{conf} =$$
$$\textbf{rec} :: 'p\ \mathit{zmset}$$
$$\textbf{msg} :: 'w \Rightarrow 'w \Rightarrow 'p\ \mathit{zmset}\ \mathit{list}$$
$$\textbf{temp} :: 'w \Rightarrow 'p\ \mathit{zmset}$$
$$\textbf{glob} :: 'w \Rightarrow 'p\ \mathit{zmset}$$



Problem 1: Accessing global state

Finite set of workers

Partially-ordered pointstamps

All pointstamps present in the system

signed multiset

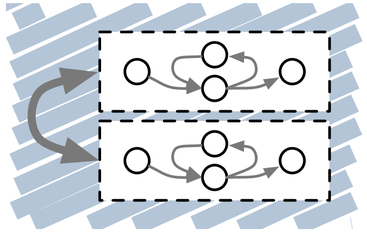
FIFO communication channels

Not-yet-communicated pointstamp updates

Each worker's local approximation of the global state

```
record ('w :: finite, 'p :: order) conf =
  rec :: 'p zmset
  msg :: 'w ⇒ 'w ⇒ 'p zmset list
  temp :: 'w ⇒ 'p zmset
  glob :: 'w ⇒ 'p zmset
```

```
definition perf_op :: 'w ⇒ 'p mset ⇒ 'p mset ⇒ ('w, 'p) conf ⇒ ('w, 'p) conf ⇒ bool where
  perf_op w Δneg Δpos c c' = let Δ = Δpos − Δneg in (∀p. Δneg#p ≤ rec c#z p) ∧ upright Δ ∧
    c' = c(|rec = rec c + Δ, temp = (temp c)(w := temp c w + Δ)|)
```



Problem 1: Accessing global state

Finite set of workers

Partially-ordered pointstamps

All pointstamps present in the system

signed multiset

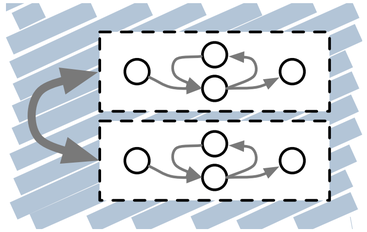
FIFO communication channels

Not-yet-communicated pointstamp updates

Each worker's local approximation of the global state

record $('w :: finite, 'p :: order)\ conf =$
  rec :: $'p\ zmset$
  msg :: $'w \Rightarrow 'w \Rightarrow 'p\ zmset\ list$
  temp :: $'w \Rightarrow 'p\ zmset$
  glob :: $'w \Rightarrow 'p\ zmset$

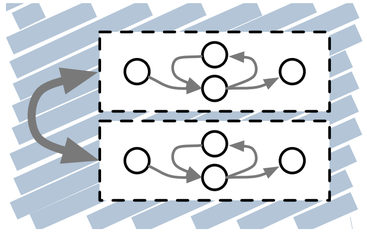
definition perf_op :: $'w \Rightarrow 'p\ mset \Rightarrow 'p\ mset \Rightarrow ('w, 'p)\ conf \Rightarrow ('w, 'p)\ conf \Rightarrow bool$ **where**
  perf_op $w\ \Delta_{neg}\ \Delta_{pos}\ c\ c' =$ let $\Delta = \Delta_{pos} - \Delta_{neg}$ in $(\forall p.\ \Delta_{neg} \# p \leqslant \mathsf{rec}\ c \#_z p) \wedge$ upright $\Delta \wedge$
  $c' = c(|\mathsf{rec} = \mathsf{rec}\ c + \Delta,\ \mathsf{temp} = (\mathsf{temp}\ c)(w := \mathsf{temp}\ c\ w + \Delta)|)$



Our exchange protocol

Addresses both problems:

1. Split global **rec** into worker-local *capabilities* **caps** :: $'w \Rightarrow 'p\ zmset$
2. Allow creation of pointstamps based on capabilities



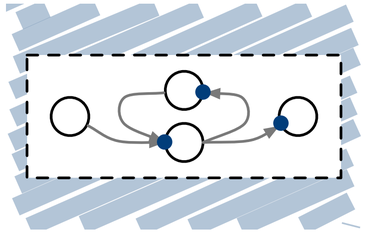
Our exchange protocol

Addresses both problems:

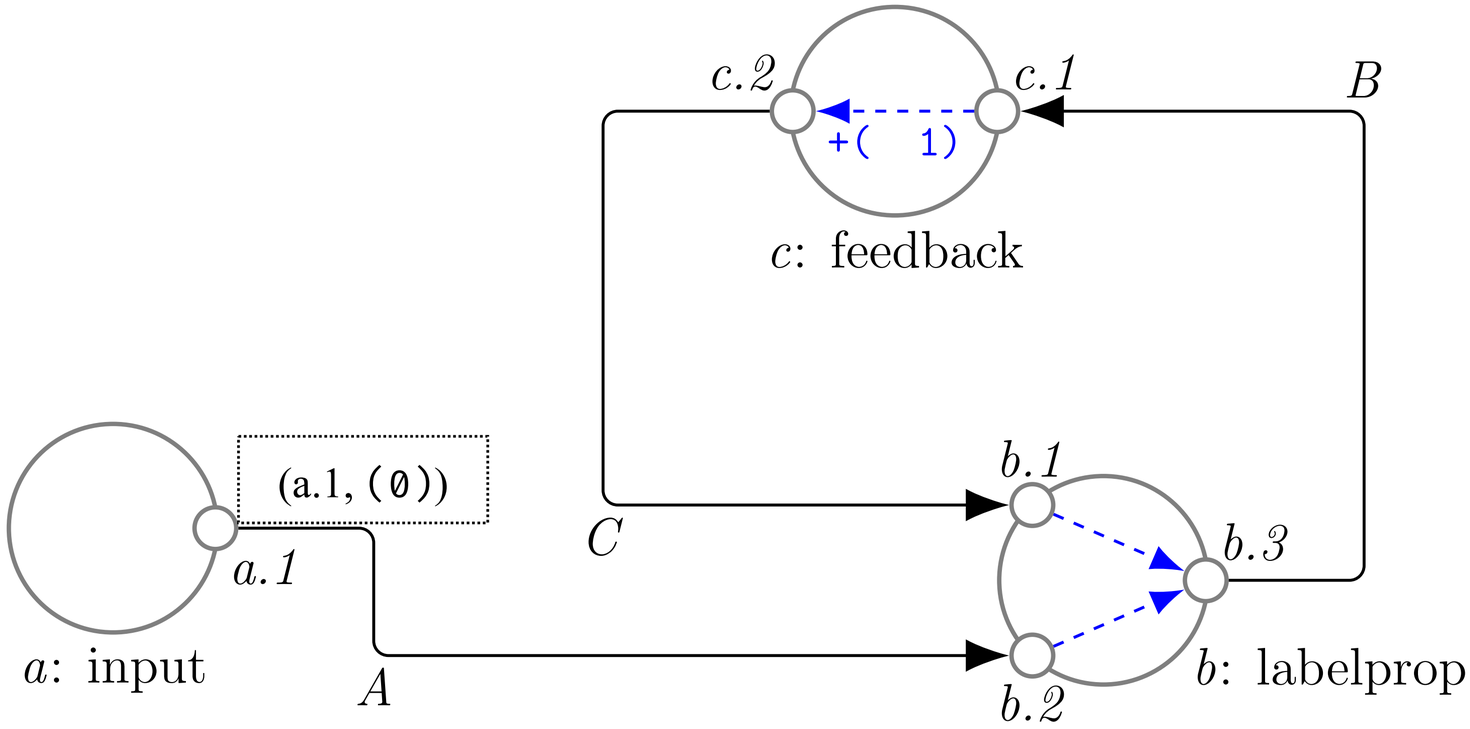
1. Split global **rec** into worker-local *capabilities* $\textbf{caps} :: {'w} \Rightarrow {'p\ zmset}$
2. Allow creation of pointstamps based on capabilities

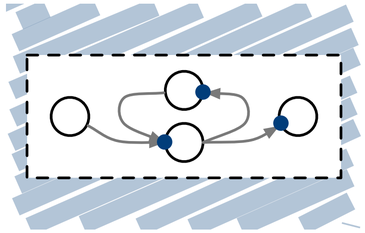
We verify the **same** safety property as Abadi et al.

If any worker's **glob** becomes vacant up to some pointstamp, then that pointstamp and any lesser ones do not exist in the system, i.e., are not present in **rec** (and will remain so).



Worker-local propagation



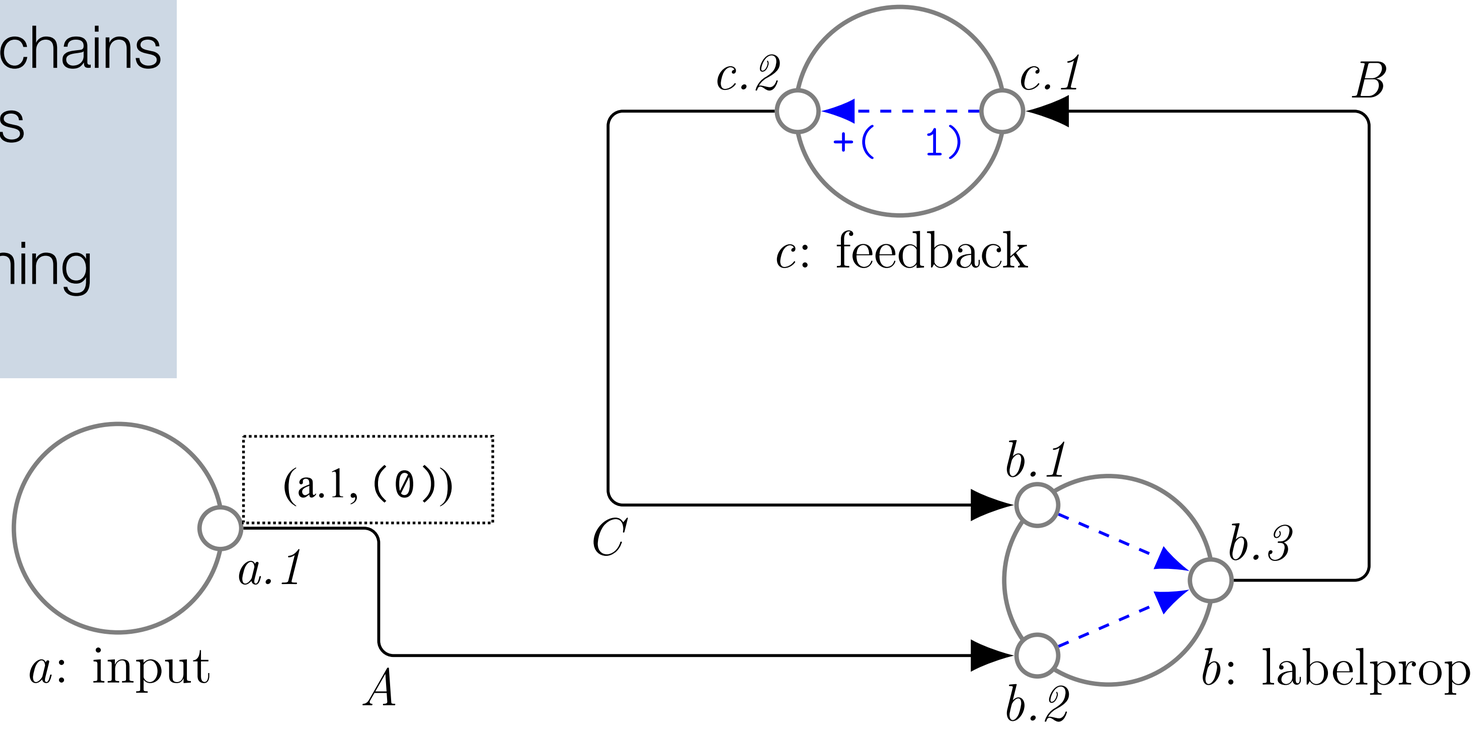


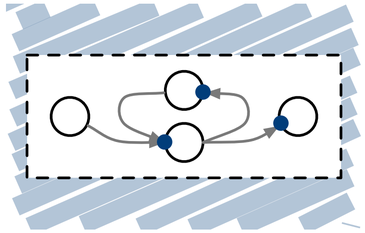
Worker-local propagation

Dataflow ≈ directed graph with antichains (over a partial order) as weights

+

A well-behaved operation combining timestamps and summaries



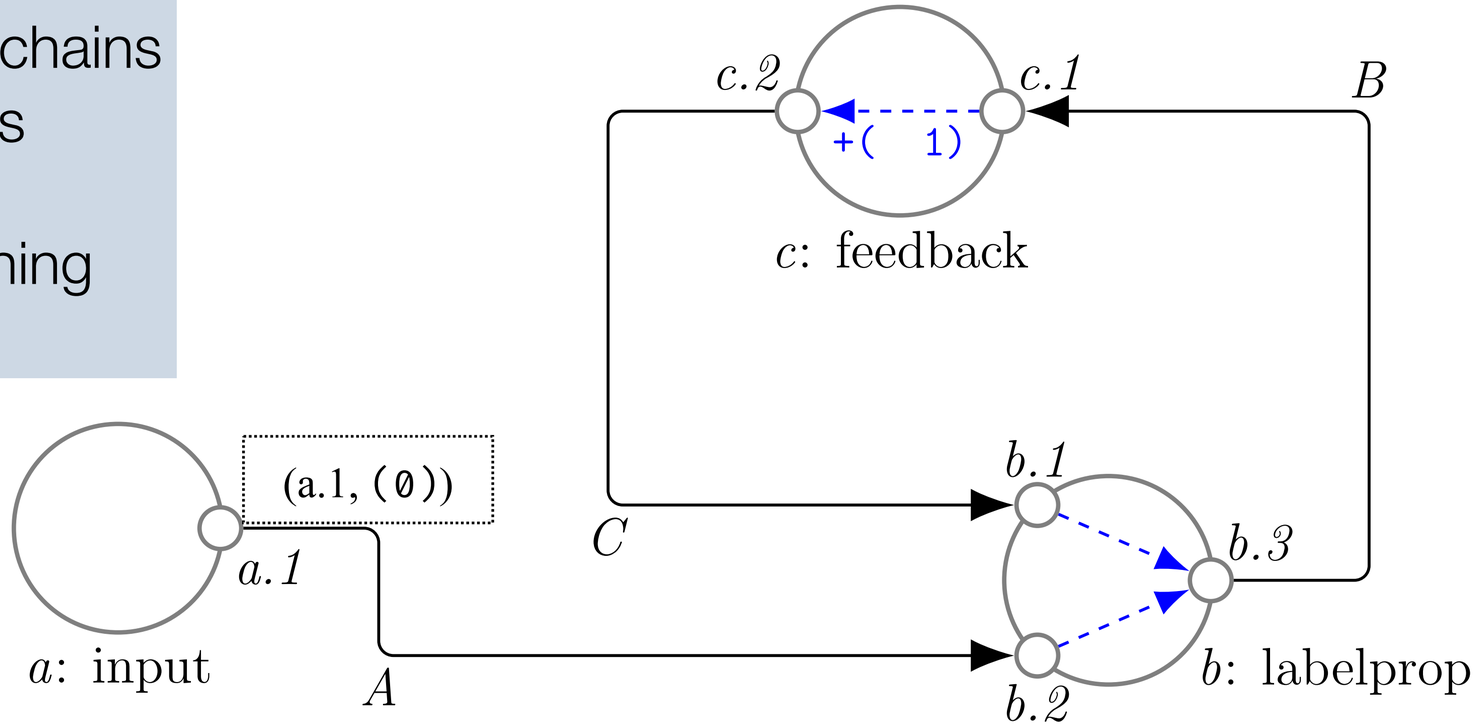


Worker-local propagation

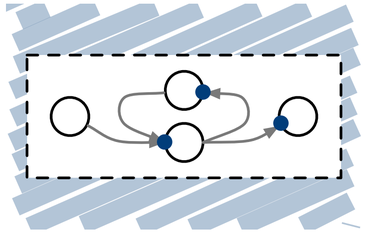
Dataflow ≈ directed graph with antichains (over a partial order) as weights

+

A well-behaved operation combining timestamps and summaries



***frontier* of a.1 is (0)**

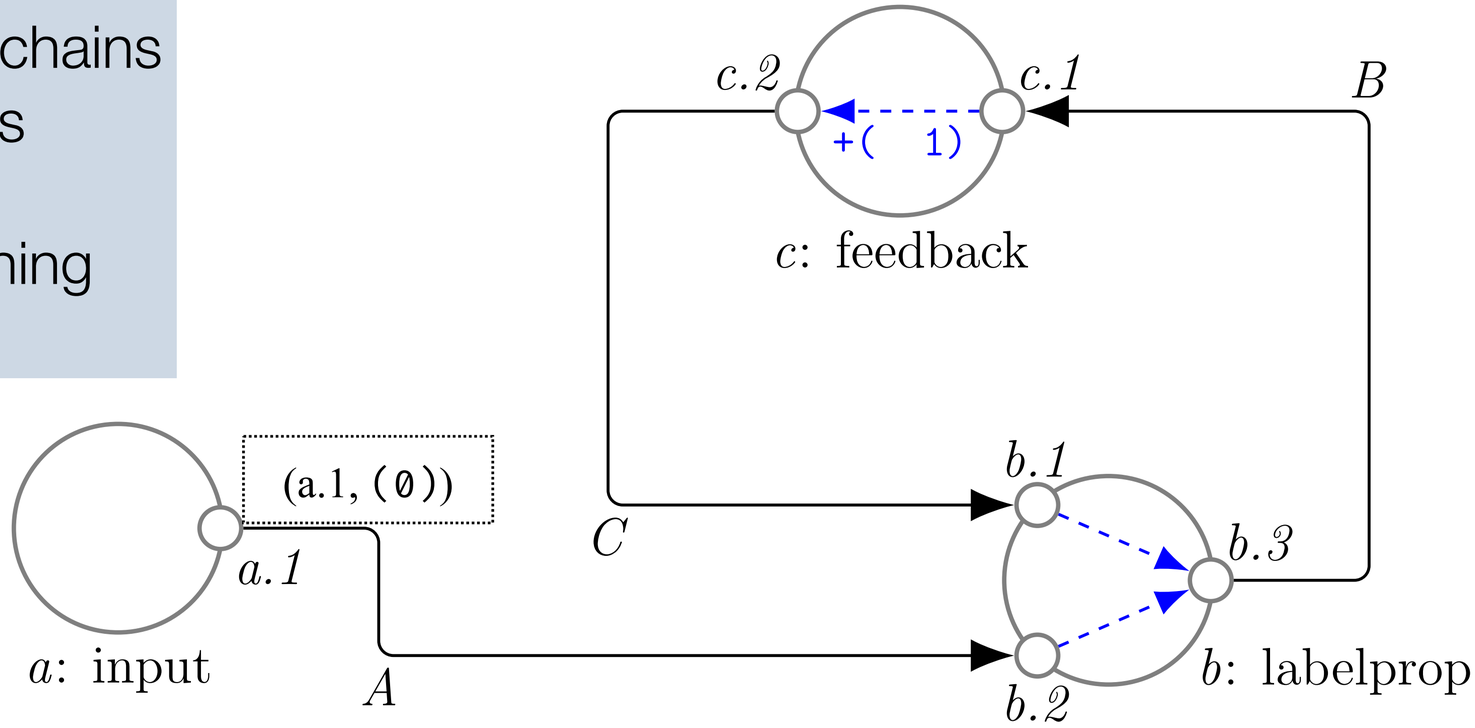


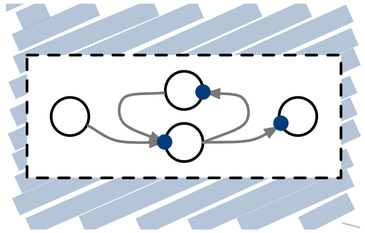
Worker-local propagation

Dataflow ≈ directed graph with antichains (over a partial order) as weights

+

A well-behaved operation combining timestamps and summaries



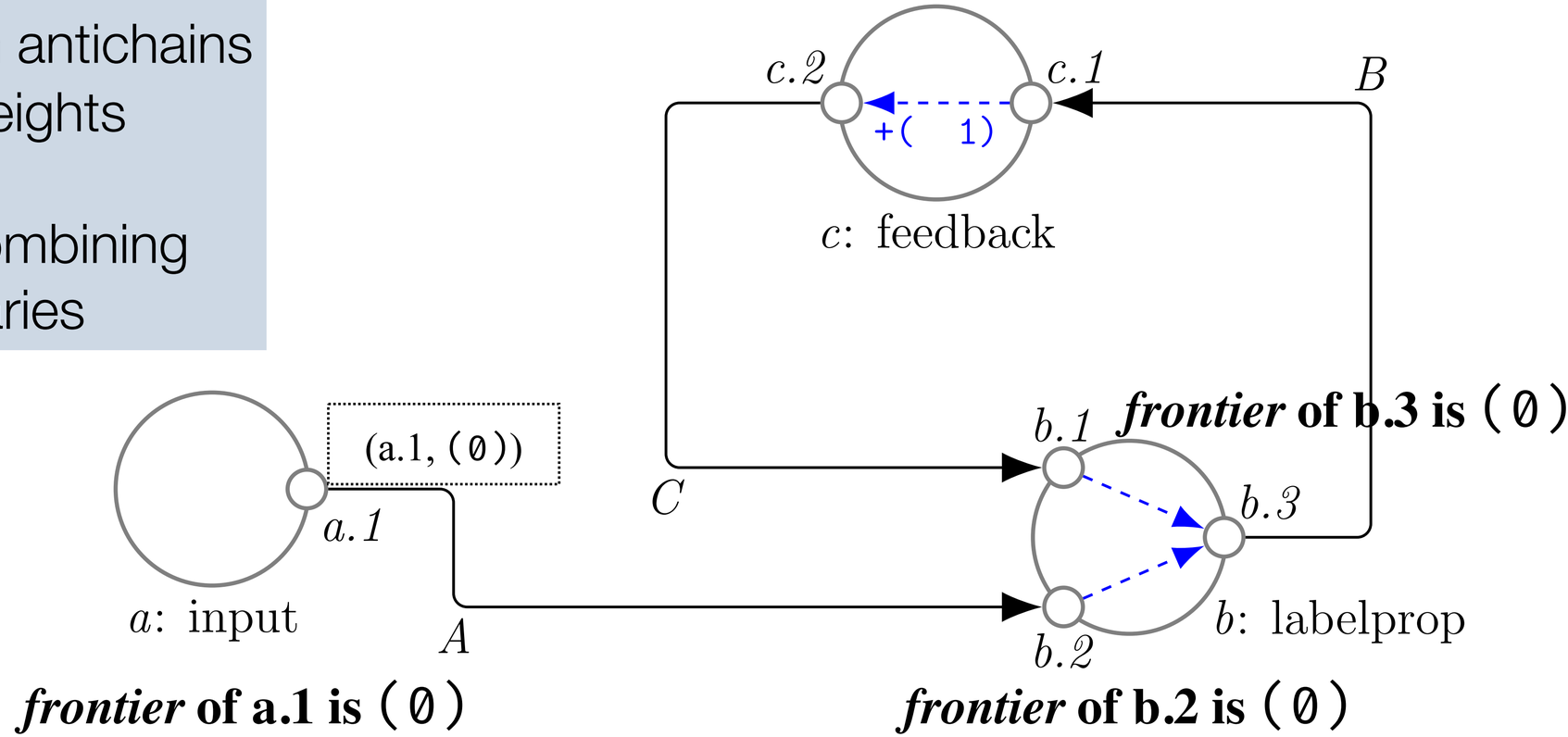


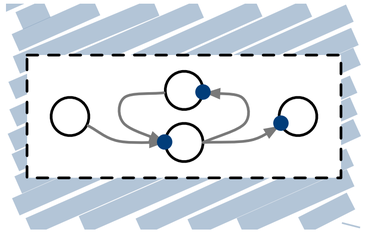
Worker-local propagation

Dataflow ≈ directed graph with antichains (over a partial order) as weights

+

A well-behaved operation combining timestamps and summaries



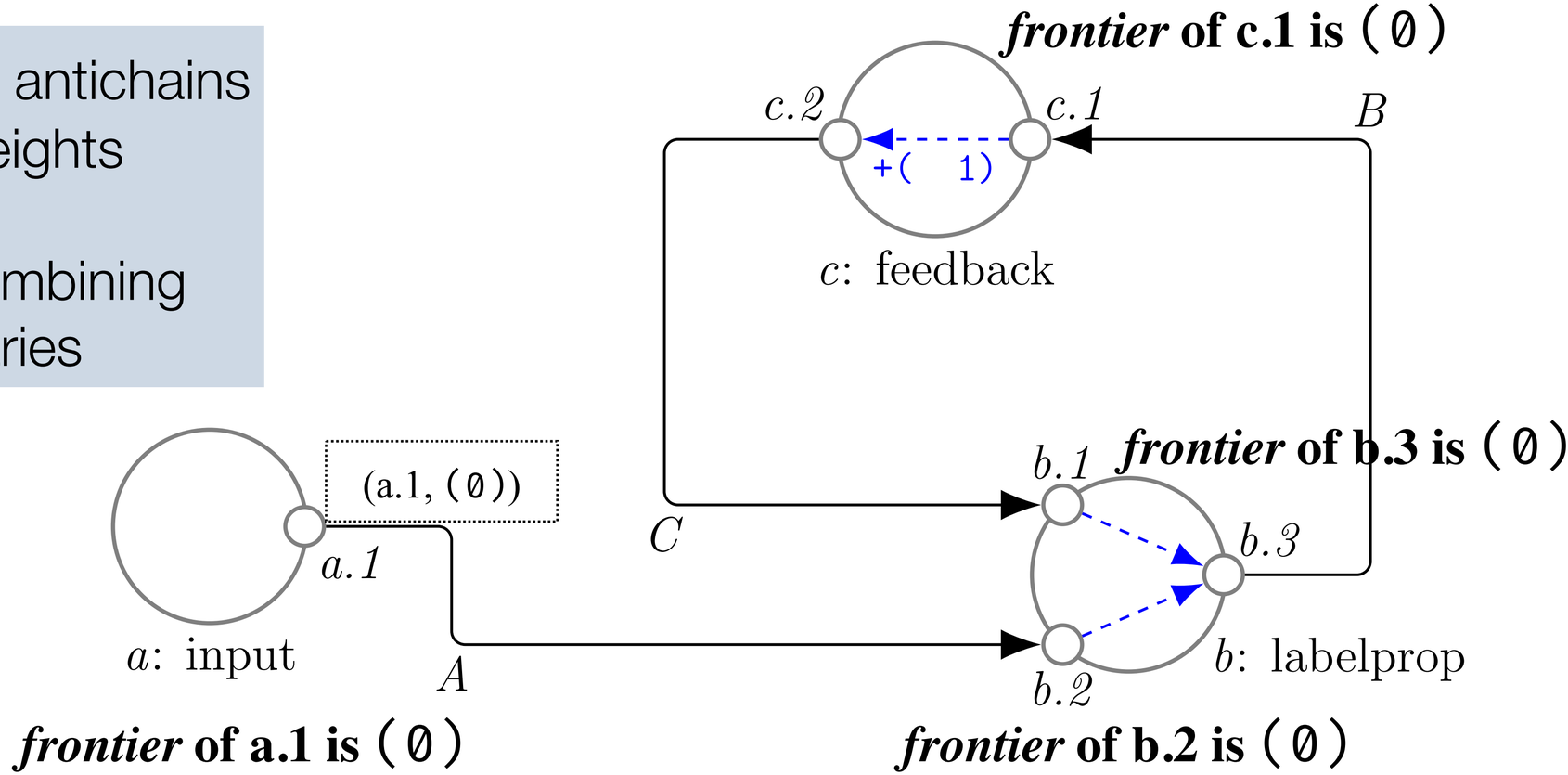


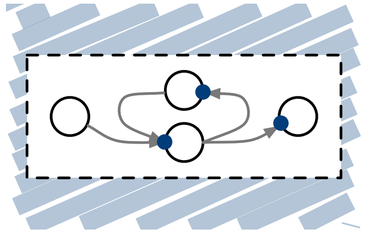
Worker-local propagation

Dataflow ≈ directed graph with antichains (over a partial order) as weights

+

A well-behaved operation combining timestamps and summaries



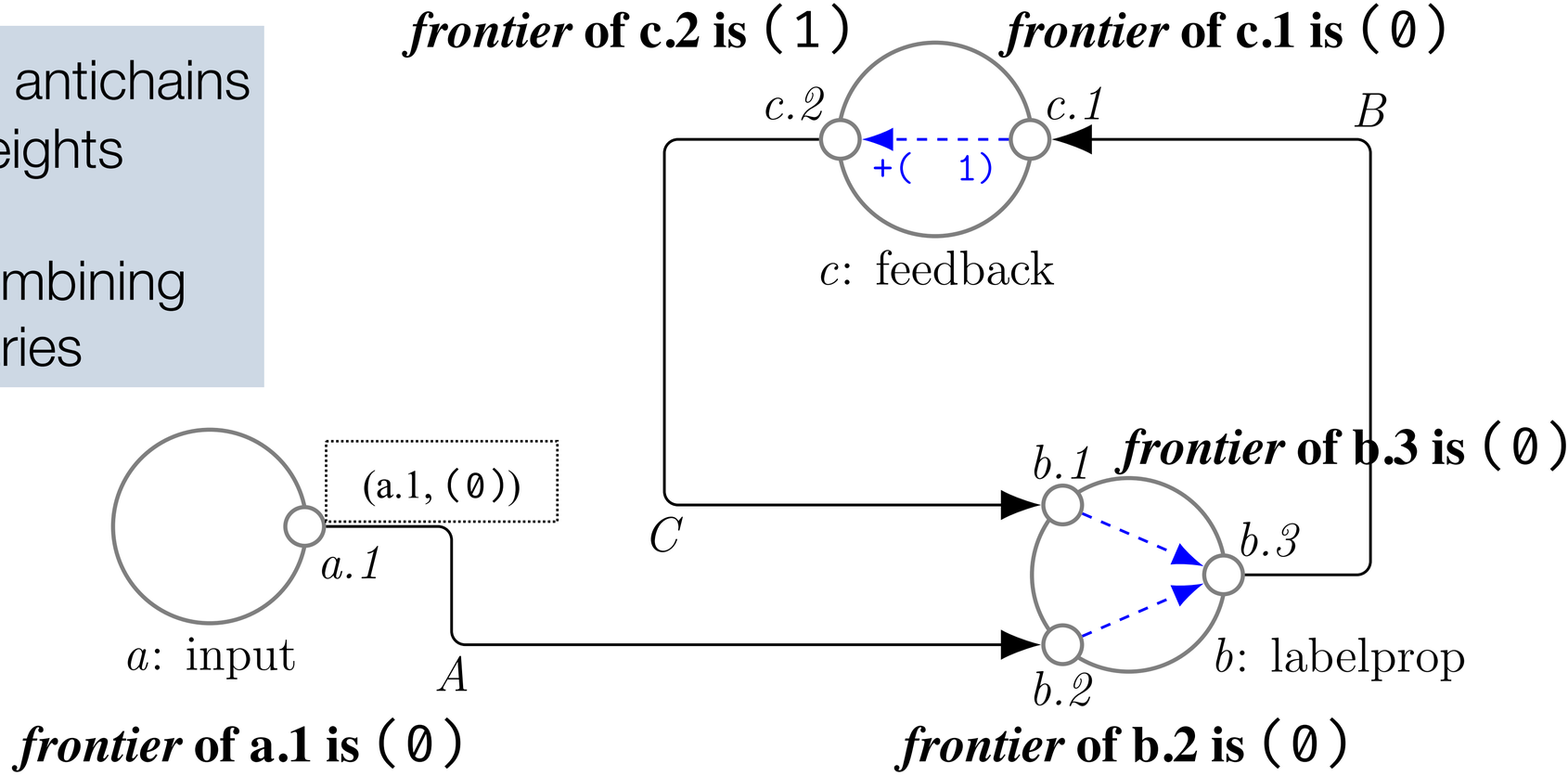


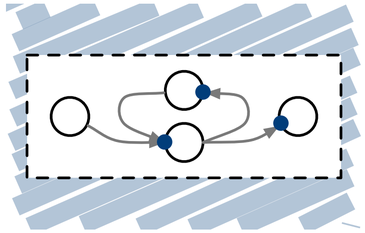
Worker-local propagation

Dataflow ≈ directed graph with antichains (over a partial order) as weights

+

A well-behaved operation combining timestamps and summaries



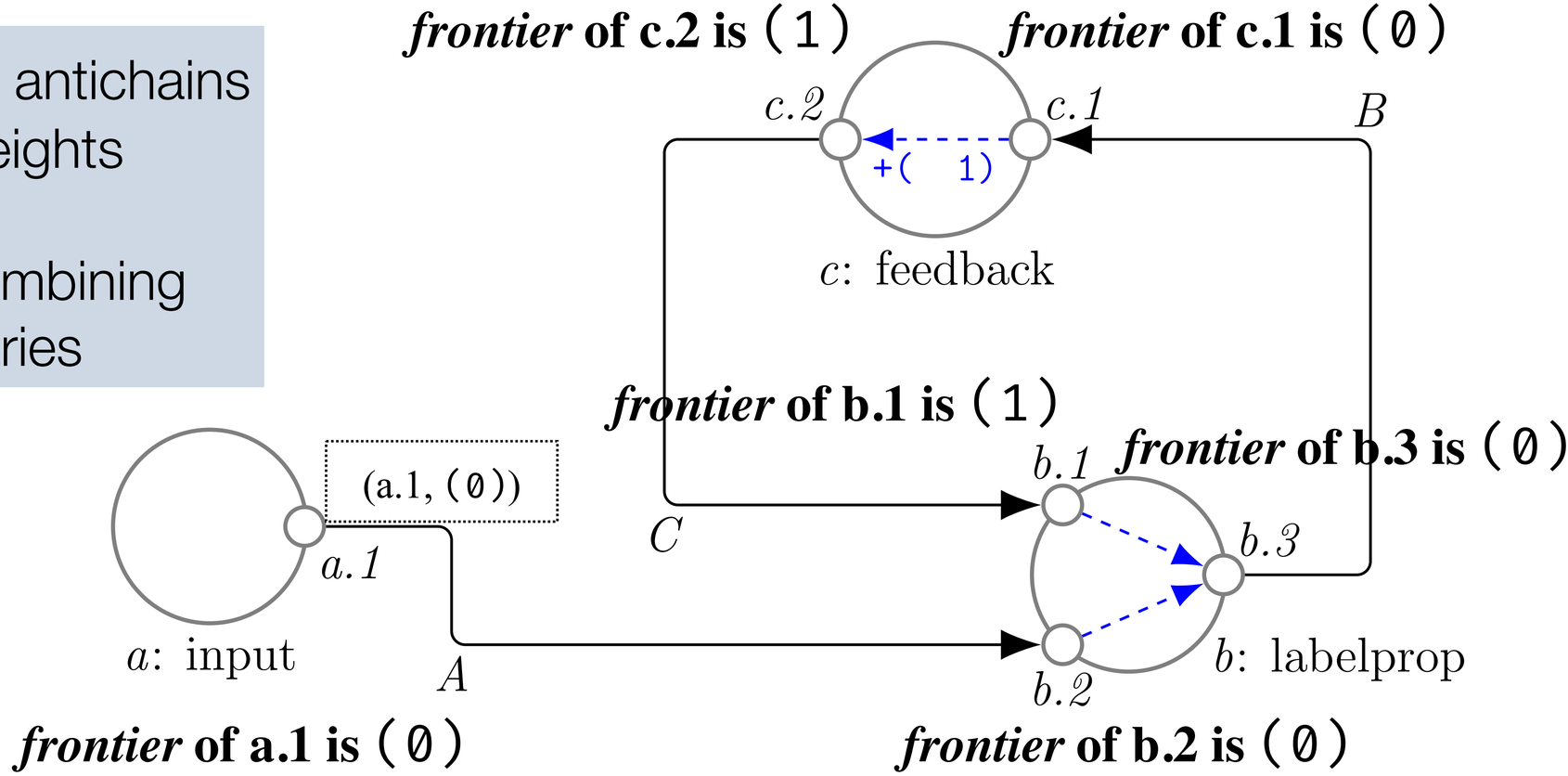


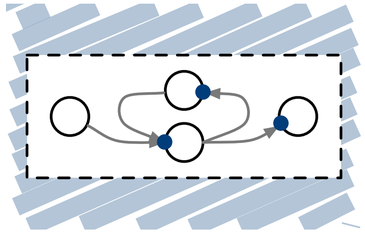
Worker-local propagation

Dataflow ≈ directed graph with antichains (over a partial order) as weights

+

A well-behaved operation combining timestamps and summaries



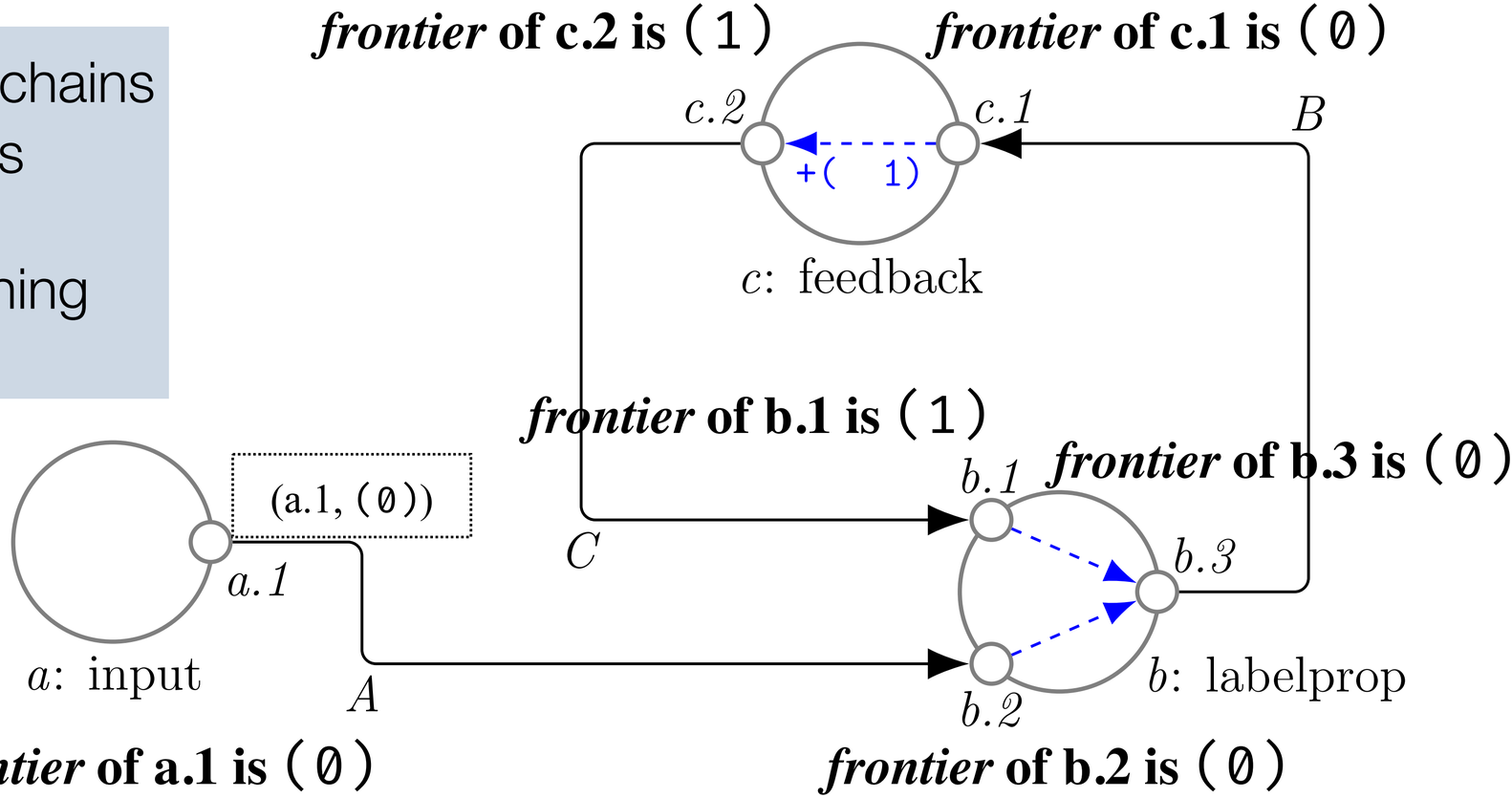


Worker-local propagation

Dataflow ≈ directed graph with antichains (over a partial order) as weights

+

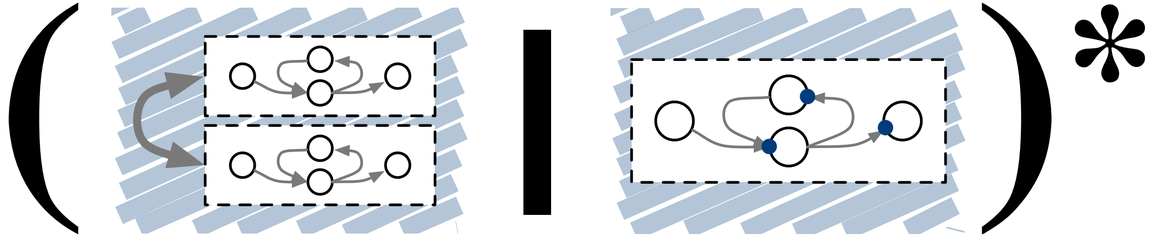
A well-behaved operation combining timestamps and summaries



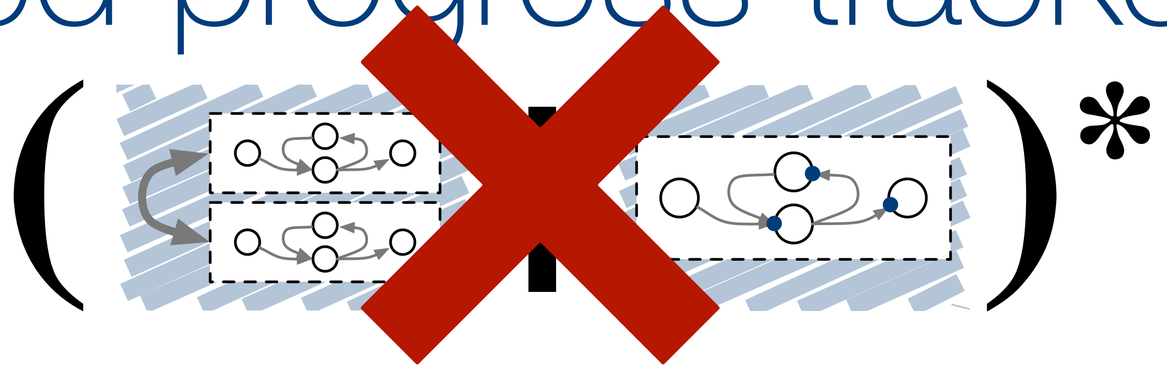
Safety property:

If all worklists neither contain timestamp ***t*** (adjusted by summaries) nor smaller timestamps, then all locations know whether they may encounter ***t*** (adjusted by summaries) in the future.

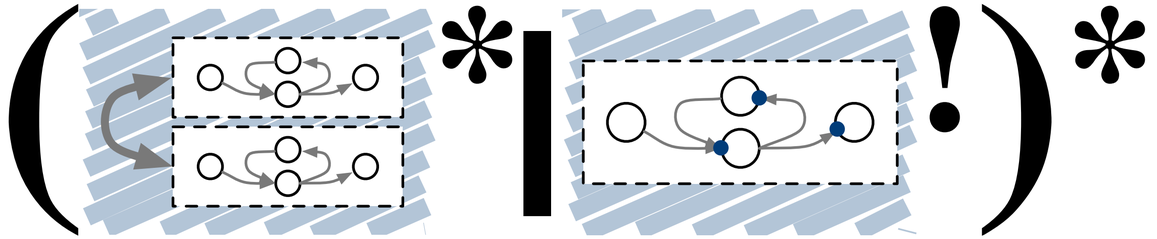
Combined progress tracker



Combined progress tracker

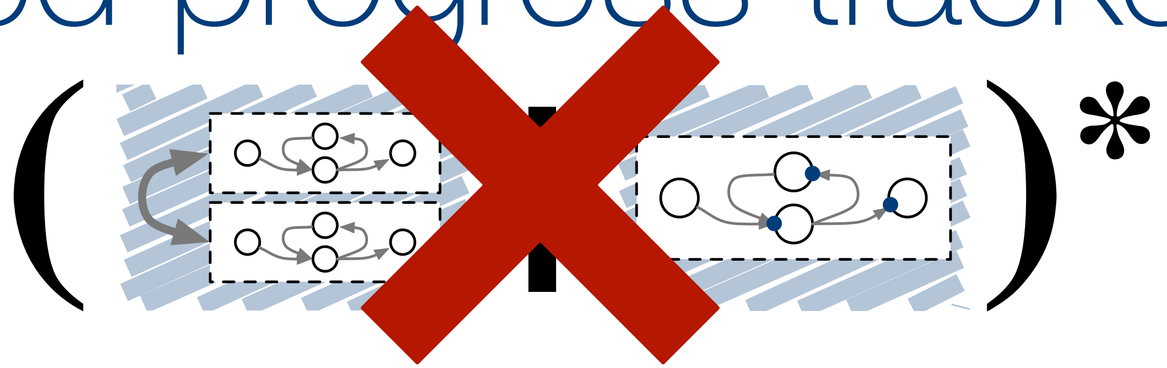


Not what Timely Dataflow does. Rather:

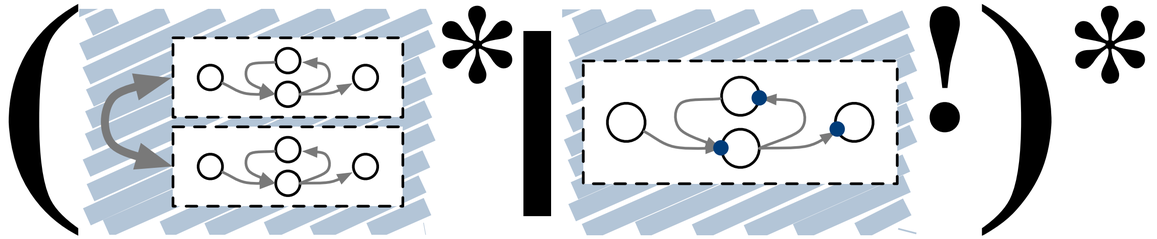


Termination of propagation not verified

Combined progress tracker



Not what Timely Dataflow does. Rather:



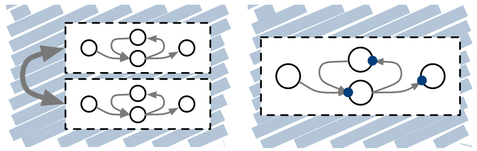
Termination of propagation not verified

Combined safety property:

Every initialized worker w has some evidence for the existence of a timestamp t at location l at *any* worker w' in w's frontier at all locations l' reachable from l.

Summary

Modeled two components

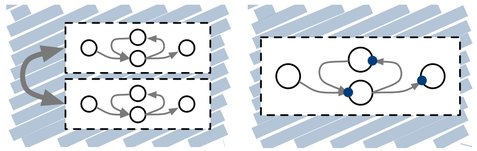


Formalized & verified safety properties for both

Verified safety of the combined Progress Tracker

https://www.isa-afp.org/entries/Progress_Tracking.html

Summary

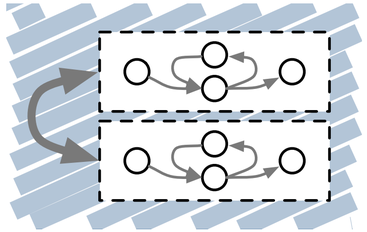
Modeled two components 

Formalized & verified safety properties for both

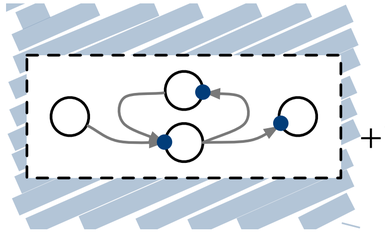
Verified safety of the combined Progress Tracker

https://www.isa-afp.org/entries/Progress_Tracking.html

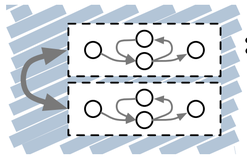
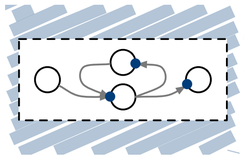
Statistics (lines of Isabelle)



3100
vs. 1000
(Abadi et al.)



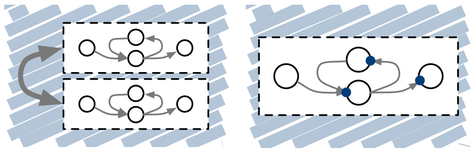
1700
+1000 library
(graphs)

( * |  !)*

1200

Summary

Modeled two components



Formalized & verified safety properties for both

Verified safety of the combined Progress Tracker

https://www.isa-afp.org/entries/Progress_Tracking.html

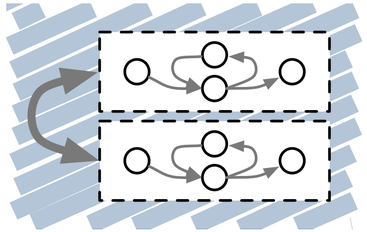
Proof assistant technology

Locales & type classes

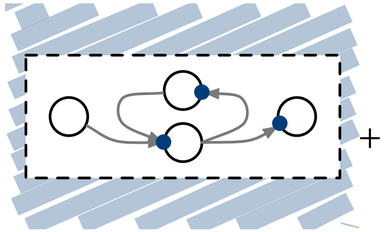
Sledgehammer

Signed Multisets

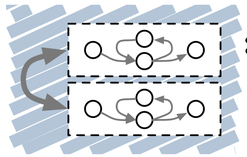
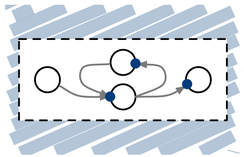
Statistics (lines of Isabelle)



3100
vs. 1000
(Abadi et al.)



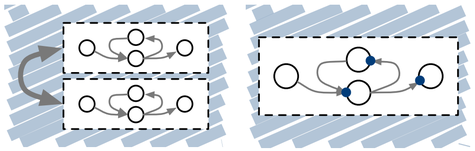
1700
+1000 library
(graphs)

( * |  !)*

1200

Summary

Modeled two components



Formalized & verified safety properties for both

Verified safety of the combined Progress Tracker

https://www.isa-afp.org/entries/Progress_Tracking.html

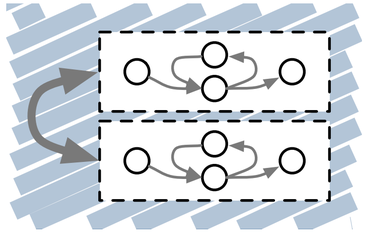
Proof assistant technology

Locales & type classes

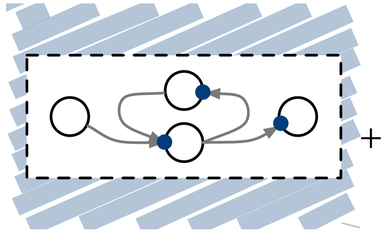
Sledgehammer

Signed Multisets

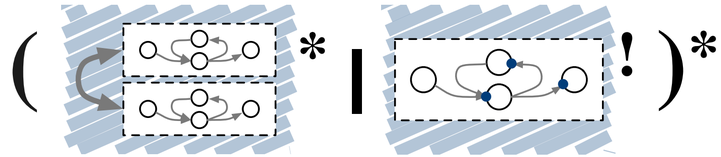
Statistics (lines of Isabelle)



3100
vs. 1000
(Abadi et al.)



1700
+1000 library
(graphs)



1200

Future Work

Termination Proof

Comparative Testing

Verify dataplane (and all of Timely)

UNIVERSITY OF COPENHAGEN

ETH zürich

Verified Progress Tracking for Timely Dataflow

Matthias Brun, Sára Decova, Andrea Lattuada
Department of Computer Science, ETH Zürich

Dmitriy Traytel
Department of Computer Science, University of Copenhagen

UNIVERSITY OF COPENHAGEN

ETH zürich

Verified Progress Tracking for Timely Dataflow

Matthias Brun, Sára Decova, Andrea Lattuada
Department of Computer Science, ETH Zürich

Dmitriy Traytel
Department of Computer Science, University of Copenhagen

Thank you!
Questions?