Concurrent Double-Ended Priority Queues

PANAGIOTA FATOUROU
FORTH ICS &
UNIVERSITY OF CRETE,
GREECE

ERIC RUPPERT
YORK UNIVERSITY,
CANADA

IOANNIS XIRADAKIS
FORTH ICS &
UNIVERSITY OF
CRETE, GREECE

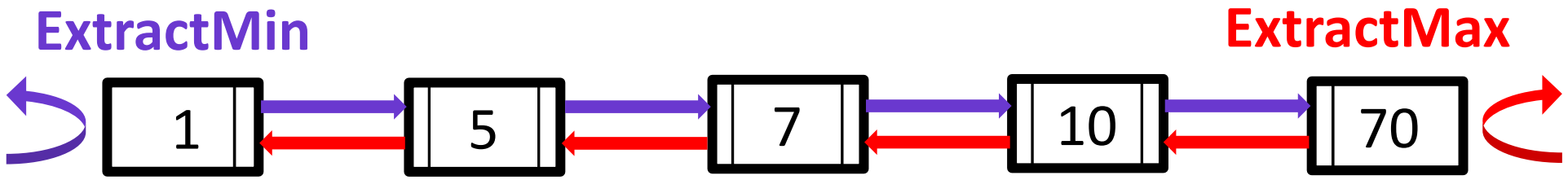
DISC 2025

* Research funded by project HAR.S.H. (project no. ΥΠ3ΤΑ-0560901), which is carried out within the framework of the National Recovery and Resilience Plan "Greece 2.0" with funding from the European Union – NextGenerationEU.

Double Ended Priority Queues (DEPQs)

A DEPQ stores a set of keys and supports the following operations:

- **Insert:** Inserts an element into the set.
- **ExtractMin:** Extracts and returns the minimum element of the set.
- **ExtractMax:** Extracts and returns the maximum element of the set.



Example of concurrent DEPQ implemented as a sorted doubly-linked list.

Applications of Priority Queues

They have been used in the following settings:

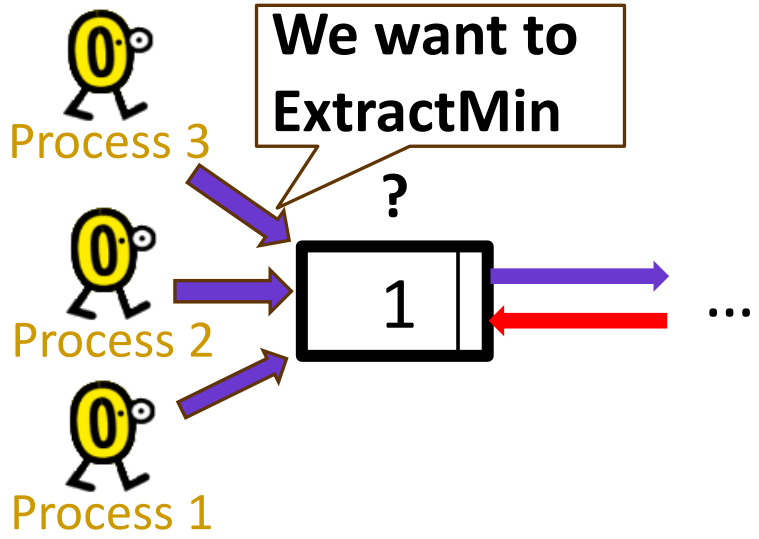
- Operating systems
- Graph algorithms
- Event driven simulation

Our Contribution

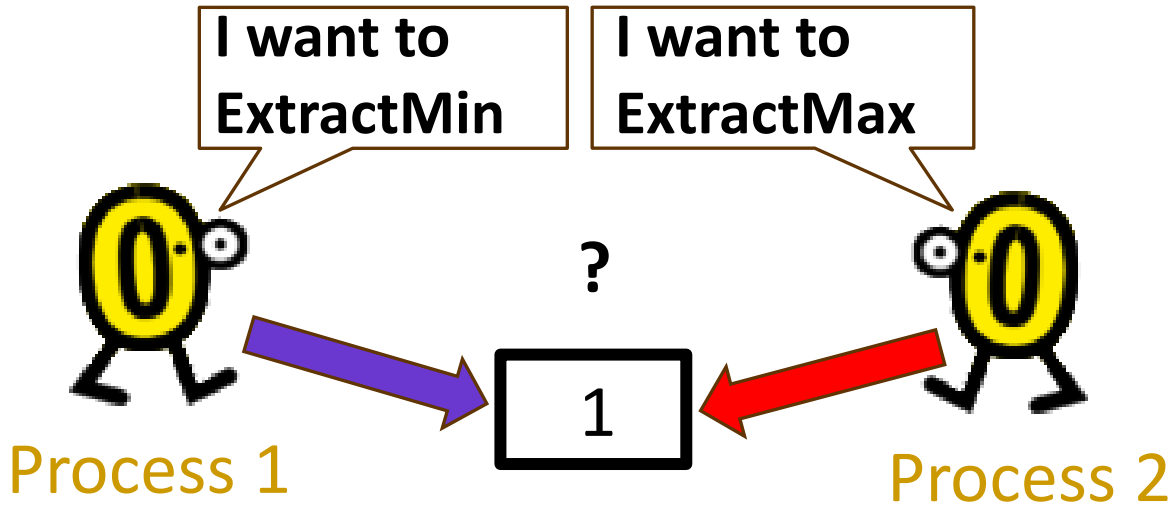
- General transformation to construct a linearizable concurrent DEPQ from a concurrent (single-ended) priority queue.
  - Insertions proceed concurrently with one another and with Extract operations.
- We use it to build the first concurrent DEPQs.
- It works even if the single-ended priority queues used are *single-consumer*, meaning that only one process at a time may perform ExtractMin operations.
- It preserves linearizability.

Concurrent Double Ended Priority Queues

- Synchronization is required between operations that perform on the same end.

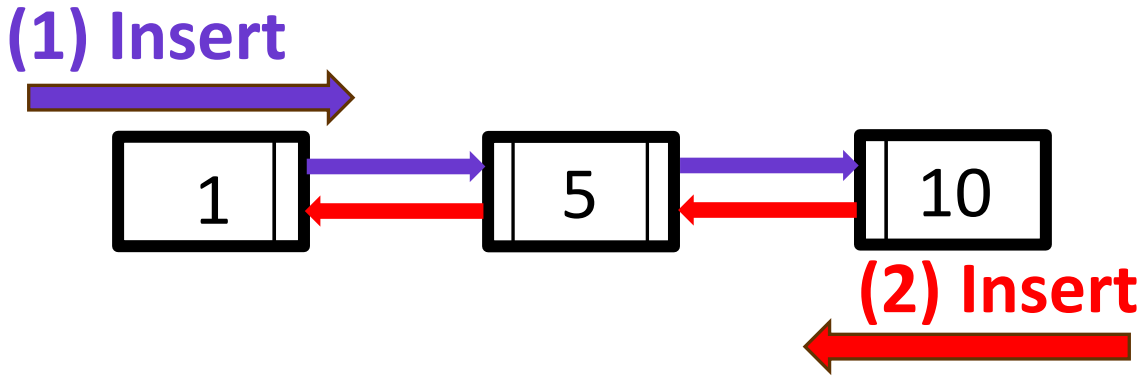


- Synchronization is required between operations that perform on different end.



Main Ideas

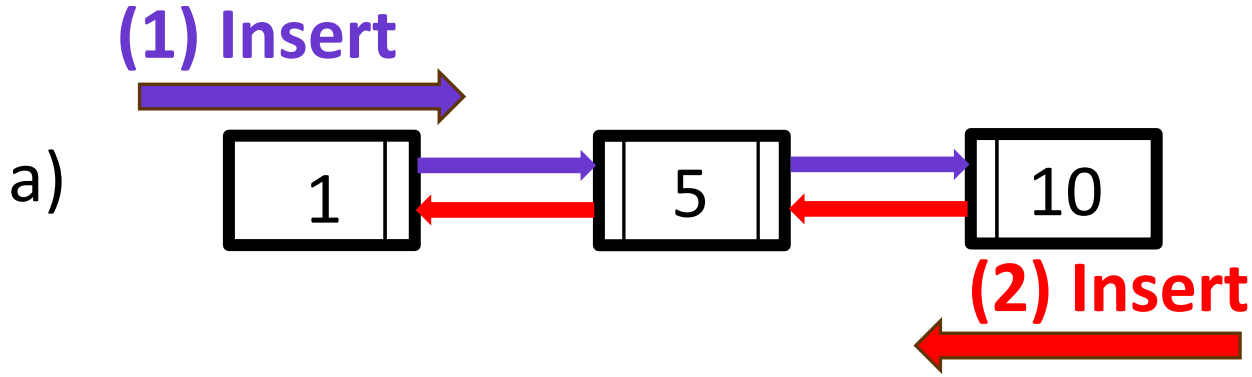
- Two Single-Ended Priority Queue instances sharing the same nodes.



Example of concurrent DEPQ implemented as two sorted single-linked list.

- Ascending order
- Descending order

FRX Technique: Insert Operation

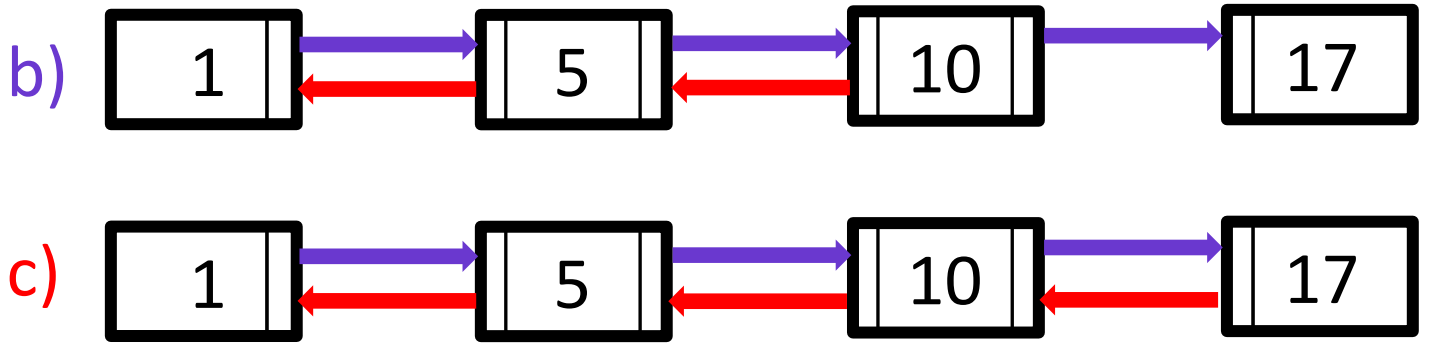


DEPQ-Insert(e)

Step 1: Call Insert(e) on min single-ended priority queue.

Step 2: Call Insert(e) on max single-ended priority queue.

DEPQ-Insert(17)



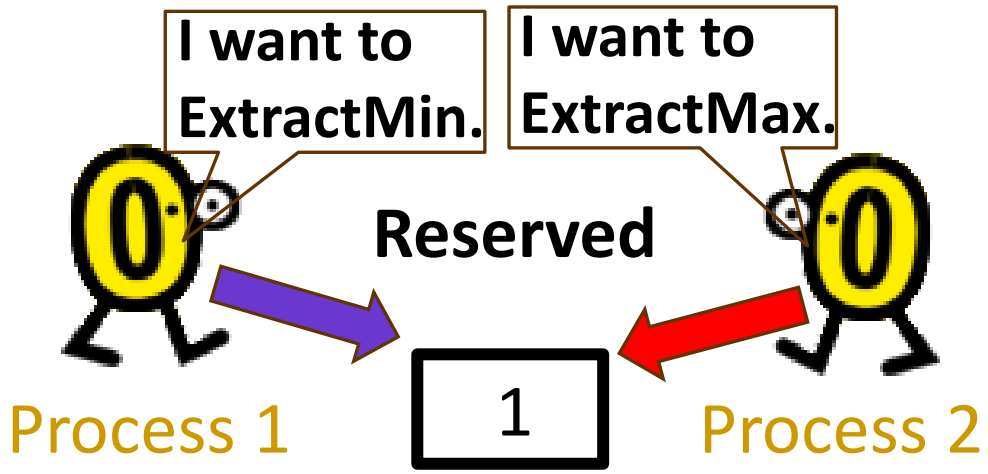
FRX Technique: Extract Operation

Each node stores a bit, called Reserved.

ExtractMin

- Repeatedly do the following:
  - Invoke ExtractMin on min priority queue; let v be the node returned.
  - Apply a Test&Set on the Reserved bit of v.
  - If Test&Set was successful
    - Delete v // if Delete is supported on single-ended PQ.
    - Return v.key

ExtractMax works in a symmetric way.



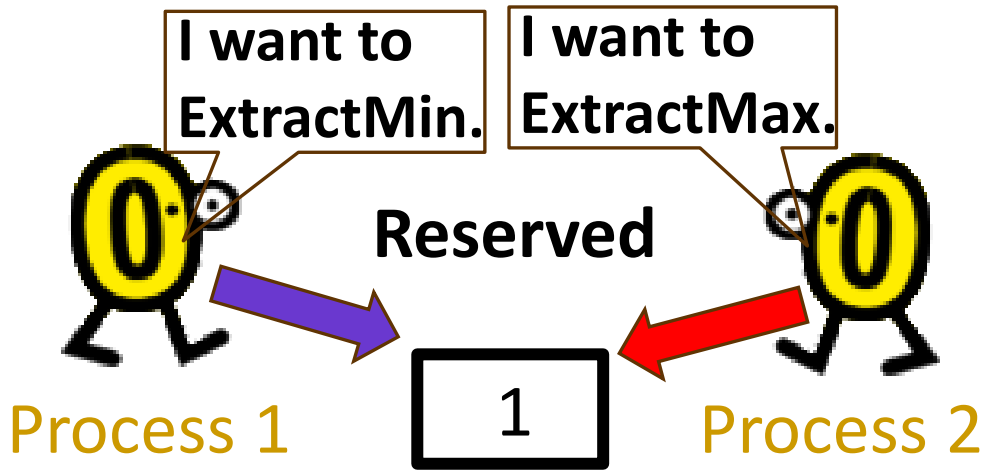
FRX Technique: Extract Operation

Each node stores a bit, called Reserved.

ExtractMin

- Repeatedly do the following:
  - Invoke ExtractMin on min priority queue; let v be the node returned.
  - Apply a Test&Set on the Reserved bit of v.
  - If Test&Set was successful
    - Delete v // if Delete is supported on single-ended PQ.
    - Return v.key

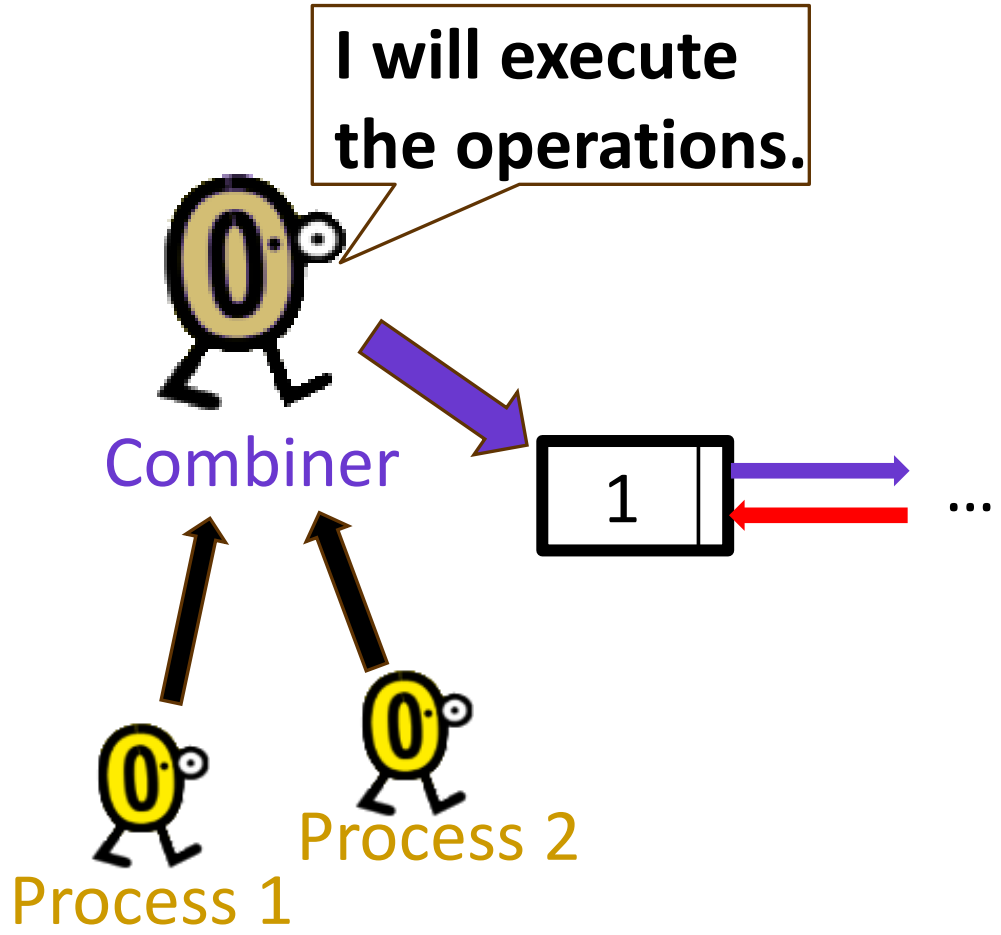
ExtractMax works in a symmetric way.



Constructing a Multi-Consumer DEPQ from a Dual-Consumer DEPQ

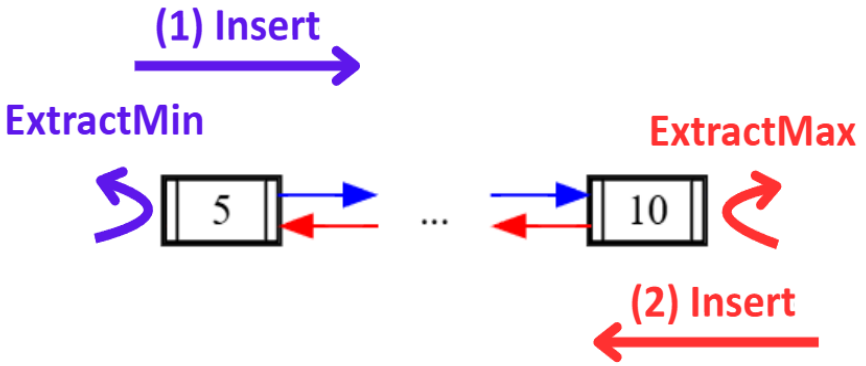
We employ CC-synch, the state-of-the-art software combining algorithm. [Fatourou & Kallimanis, PPoPP 2022]

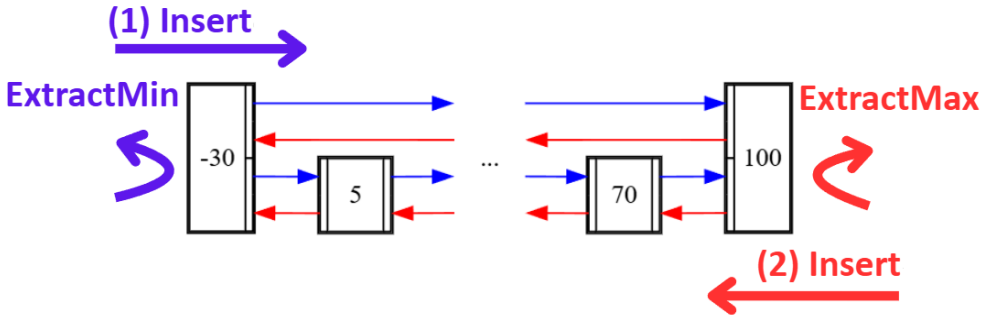
- One combiner on each end that undertakes the task to execute the Extract operations invoked by the threads on that end.



Conclusion

- General transformation to construct linearizable concurrent DEPQs from concurrent single-ended PQs.
- We utilize the construction to get the first concurrent DEPQs.
- Recent experimental work shows that applying the technique to various existing single-ended priority queues constructs fast linearizable concurrent DEPQs.





Thank you!

Ioannis Xiradakis
giannisx@ics.forth.gr

HAR.S.H. HARdware-aware Similarity searcH
Greece 2.0 NATIONAL RECOVERY AND RESILIENCE PLAN
Funded by the European Union NextGenerationEU

Constructing a Multi-Consumer DEPQ from a Dual-Consumer DEPQ

We employ CC-synch, the state-of-the-art software combining algorithm.[Fatourou & Kallimanis, PPoPP 2022]

Main Ideas in Software Combining

- Processes first announce their synchronization requests by e.g., writing a function pointer in a shared variable.
- Processes synchronize with one another to choose a combiner process, which serves in addition to its own request, active requests by other processes.
- Processes that do not become combiners perform local spinning until the combiner informs them that it has performed their requests.