

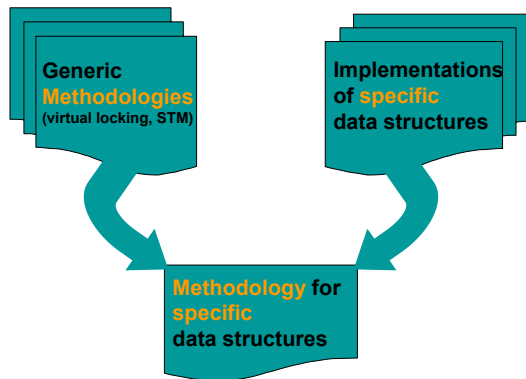
Built-in Coloring for Highly-Concurrent Doubly-Linked Lists

Hagit Attiya and Eshcar Hillel
Computer Science Department
Technion

Concurrent Data Structures

- At the core of many distributed applications
 - E.g., doubly-linked lists are used as queues
- Lock-based implementations
 - Deadlock when locking in a different order
 - Blocking when processes are slow or fail
 - Do not scale
- Lock-free implementations, esp., **nonblocking**
 - Often complex and hard to get right

Context...

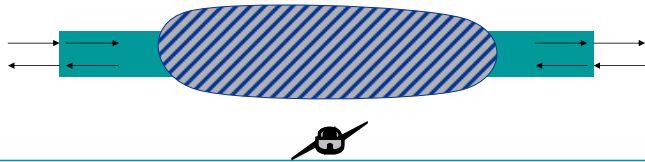


Our Contribution

- An approach for implementing specific data structures
 - Semantics-aware
- ➔ Two doubly-linked list implementations
 - Removals only at ends, using **compare&swap** (CAS)
 - Insertions & removals anywhere, using double-CAS (2CAS, **DCAS**)
 - Both algorithms are nonblocking and with little interference between operations.

Implementing by Atomically Locking Multiple Locations

- Implementing a doubly-linked list is easy if we can lock three locations atomically
- This can be simulated using (single-location) CAS:
 - Virtual locking
 - Software transactional memory
 - Color-based locking



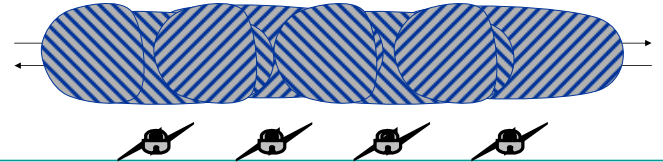
Built-in Coloring for Doubly-Linked Lists

5

Virtual Locking

[Turek, Shasha, Prakash]
[Barnes]

- Acquire **virtual locks** on nodes in the data set
- Help **blocking** operations to complete
- May result in long **helping chains**



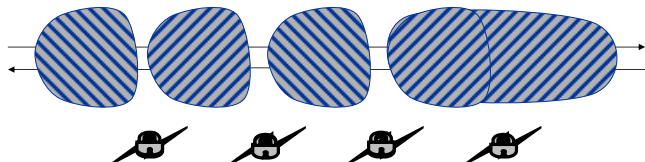
Built-in Coloring for Doubly-Linked Lists

6

Software Transactional Memory (STM)

[Shavit, Touitou]

- Release locks when the operation is blocked, and retry
- Help only an **immediate** neighbor
 - Short helping chains
 - Still may incur long delay chains



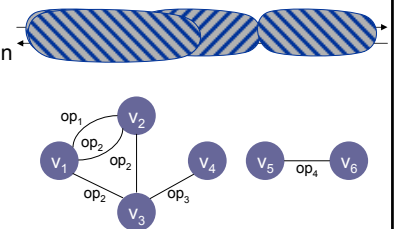
Built-in Coloring for Doubly-Linked Lists

7

Measuring Non-Interference (Locality)

[Afek, Merritt, Taubenfeld, Touitou]

- **d -local step complexity**
 - operations delay each other
- **d -local contention**
 - access same memory location
- Virtual locking:
 - n -local step complexity
 - n -local contention
- STM:
 - n -local step complexity
 - $O(1)$ -local contention



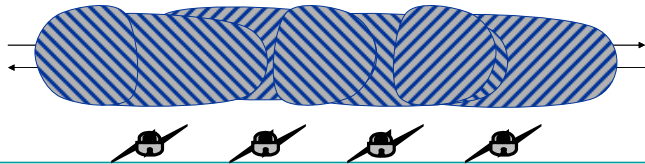
Built-in Coloring for Doubly-Linked Lists

8

Color-based Locking Scheme

[Attiya, Dagan]
[Afek, Merritt, Taubenfeld, Touitou]

- Colors define the locking order ■ < ■ < ■
- Bound the length of delay chains
- Color the items when the operation starts
 - Complicated and only \log^*n -local



Built-in Coloring for Doubly-Linked Lists

9

Why Color from Scratch?

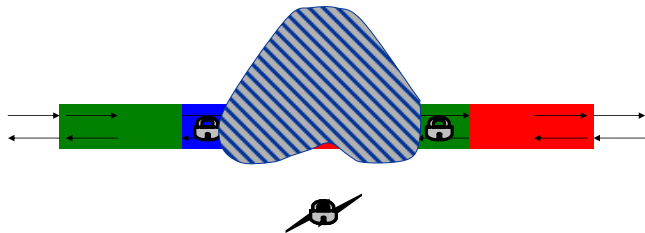
- For a specific data structure
 - We construct the linked list from its initialization...
 - Operations access the nodes in a predictable pattern
- 🔑 Maintain the nodes always legally colored while inserting & removing
- No cost for re-coloring
 - Constant locality, spec., when the linked list is not “empty”, operations on opposite ends do not delay each other
- CAS-Chromo: removals only at ends
- DCAS-Chromo: Insertions & removals anywhere

Built-in Coloring for Doubly-Linked Lists

10

Algorithm CAS-Chromo: Insert Operation

- Use temporary color < ■ < ■ < ■

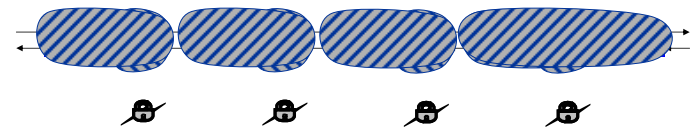


Built-in Coloring for Doubly-Linked Lists

11

Removing a Node from the Middle

- A chain of Remove operations may cause trouble

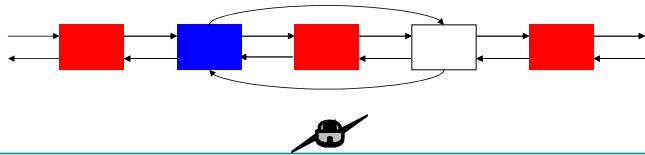


Built-in Coloring for Doubly-Linked Lists

12

Algorithm DCAS-Chromo: Remove Operation

👉 Use DCAS to lock equally colored nodes



Previous Linked List Implementations

	Primitives	Interference	Comments
Greenwald	DCAS	any pair	
Harris	CAS	any pair	Singly-linked list
Michael	CAS	any pair	Singly-linked list
Sundell and Tsigas	CAS	any pair	
DCAS-Chromo	DCAS	distance < 3	

Previous Deque Implementations

	Primitives	Interference	Comments
Michael	CAS	opposite ends	
Herlihy et al.	CAS	distance < 2	obstruction free
Greenwald	DCAS	opposite ends	
Agesen et. al.	DCAS	distance < 2	
Sundell and Tsigas	CAS	distance < 2	
CAS-Chromo	CAS	distance < 3	
CAS-Chromo	CAS	distance < 5	insertions anywhere

Conclusions & Further research

- Virtual locking with built-in coloring
 - Low interference
 - Simplifies the design
 - Provides clean functionality
 - Facilitates correctness proof
- Further research
 - Other data structures
 - Is DCAS necessary for removals from the middle?
 - Lock-based implementations
 - Constant failure locality [Choi, Singh]
 - ...

QUESTIONS?