On the design space of Parallel Nesting

Nuno Diegues João Cachopo

nmld, joao.cachopo@ist.utl.pt

INESC-ID/Technical University of Lisbon

July 19, 2012

Selling point of TM

Composability

Selling point of TM

Composability

Parallel Nesting

Time complexity analysis may be deceiving in TMs

Outline

Compare three parallel nesting approaches

1. JVSTM
2. NesTM[1]
3. PNSTM[2]

[1] W. Baek, N. Bronson, C. Kozyrakis, and K. Olukotun. Implementing and evaluating nested parallel transactions in software transactional memory. In SPAA '10.

[2] J. Barreto, A. Dragojević, P. Ferreira, R. Guerraoui, and M. Kapalka. Leveraging parallel nesting in transactional memory. In PPoPP '10.

Outline

Compare three parallel nesting approaches

1. JVSTM ←
2. NesTM[1]
3. PNSTM[2]

[1] W. Baek, N. Bronson, C. Kozyrakis, and K. Olukotun. Implementing and evaluating nested parallel transactions in software transactional memory. In SPAA '10.

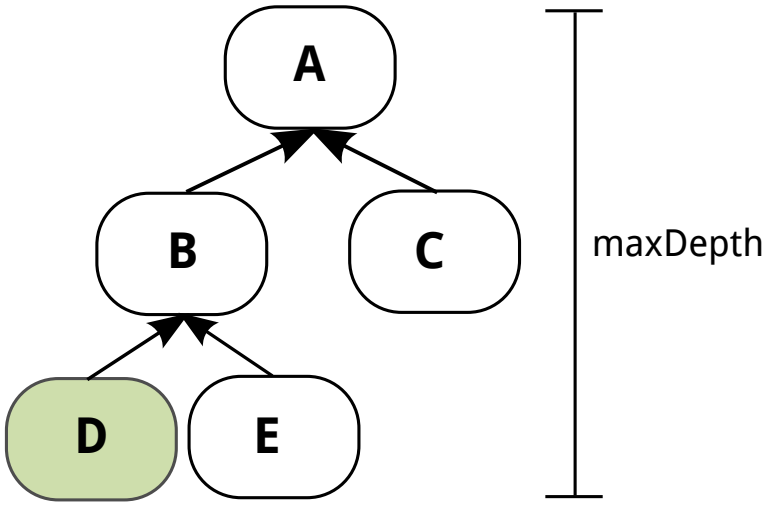
[2] J. Barreto, A. Dragojević, P. Ferreira, R. Guerraoui, and M. Kapalka. Leveraging parallel nesting in transactional memory. In PPoPP '10.

Worst-case complexities - JVSTM

	JVSTM
read	$O(maxDepth)$
write	$O(1)$
commit	$O(r + children)$

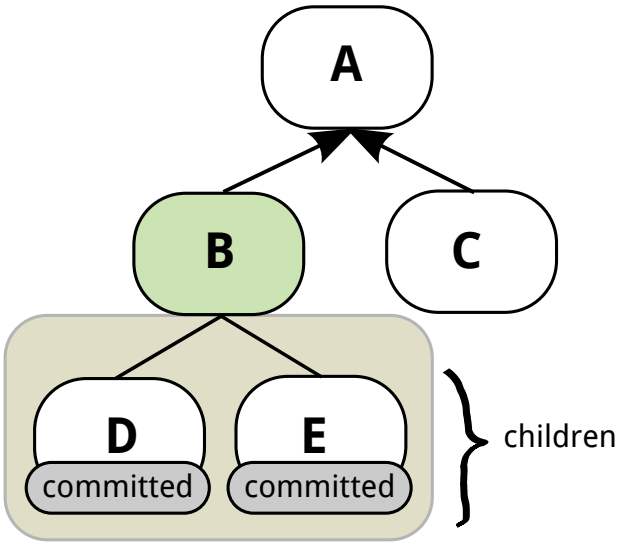
Worst-case complexities - JVSTM

	JVSTM
read	**O(maxDepth)**
write	$O(1)$
commit	$O(r + children)$



Worst-case complexities - JVSTM

	JVSTM
read	$O(maxDepth)$
write	$O(1)$
commit	**O(r+children)**

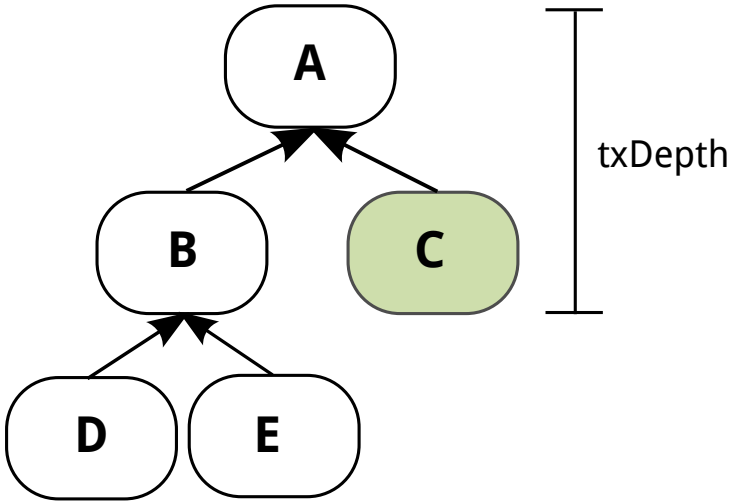


Worst-case complexities - NesTM

	JVSTM	NesTM
read	$O(maxDepth)$	$O(1)$
write	$O(1)$	$O(txDepth)$
commit	$O(r + children)$	$O(r + w)$

Worst-case complexities - NesTM

	JVSTM	NesTM
read	$O(maxDepth)$	$O(1)$
write	$O(1)$	**O(txDepth)**
commit	$O(r + children)$	$O(r + w)$



Worst-case complexities - PNSTM

	JVSTM	NesTM	PNSTM
read	$O(maxDepth)$	$O(1)$	$O(1)$
write	$O(1)$	$O(txDepth)$	$O(1)$
commit	$O(r + children)$	$O(r + w)$	$O(1)$

Worst-case complexities

	JVSTM	NesTM	PNSTM
read	$O(maxDepth)$	$O(1)$	**O(1)**
write	$O(1)$	$O(txDepth)$	**O(1)**
commit	$O(r + children)$	$O(r + w)$	**O(1)**

Best one?

Practical comparison

- STMBench7 - running given number of transactions

Practical comparison

- STMBench7 - running given number of transactions
- Implementation of STMs

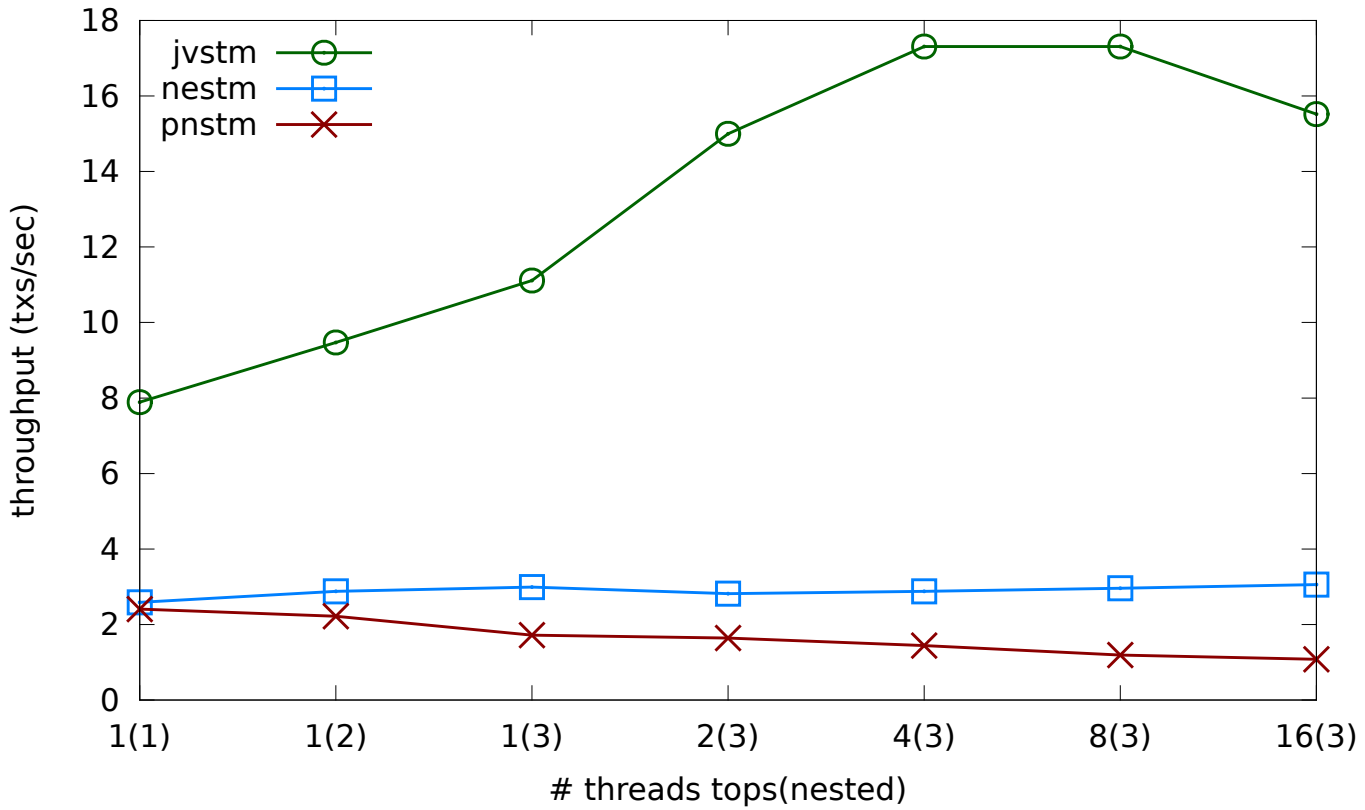
Practical comparison

- STMBench7 - running given number of transactions
- Implementation of STMs
- Same API

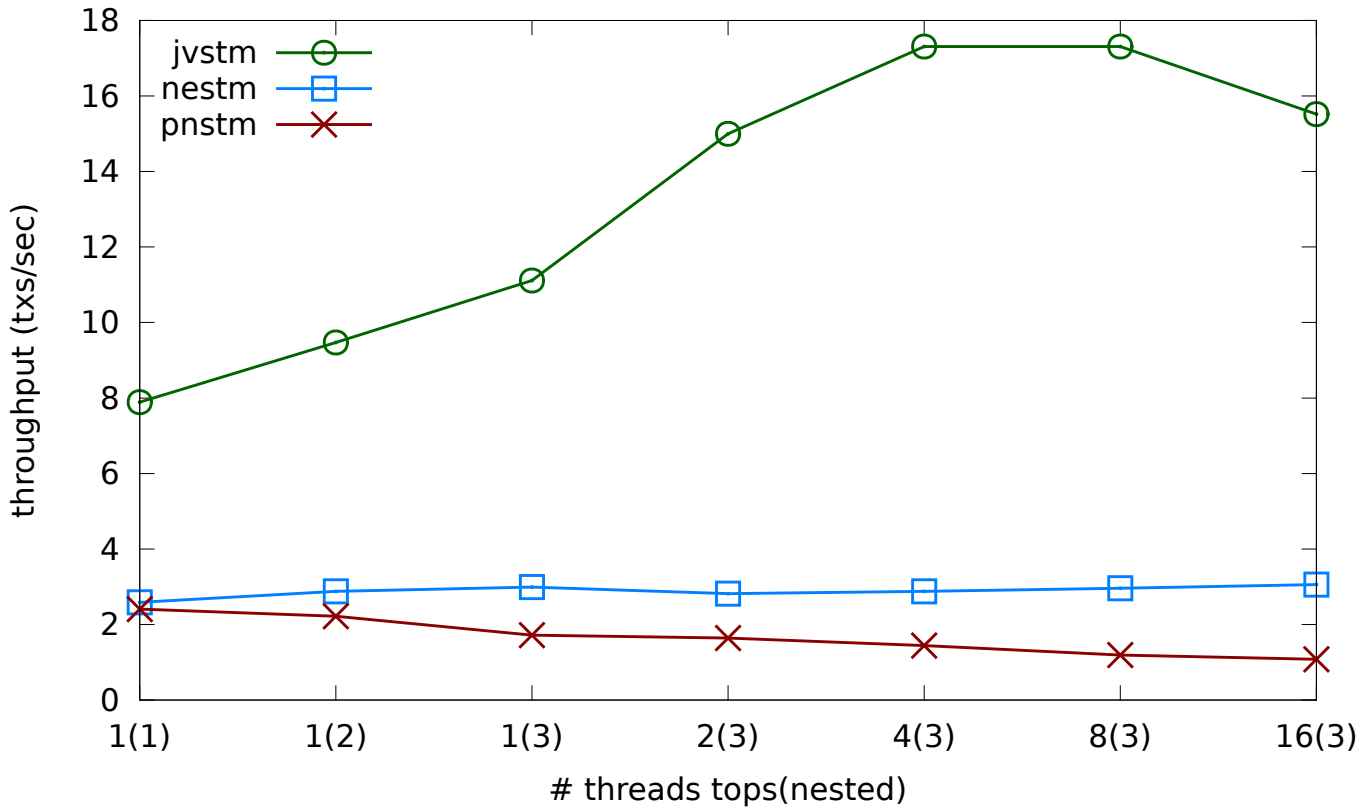
Practical comparison

- STMBench7 - running given number of transactions
- Implementation of STMs
- Same API
- 48 core machine

STMBench7

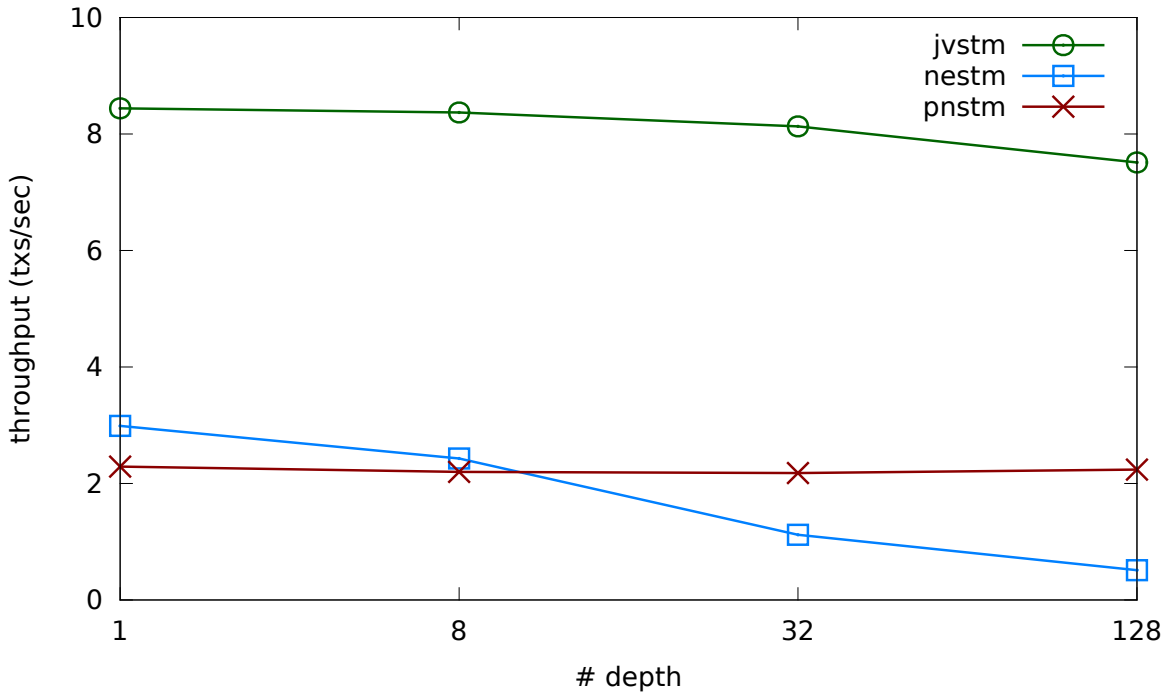

jvstm
nestm
pnstm
throughput (txs/sec)
0
2
4
6
8
10
12
14
16
18
1(1)
1(2)
1(3)
2(3)
4(3)
8(3)
16(3)
# threads tops(nested)

STMBench7



- 5 and 15 times with 48 threads/parallel nested

STMBench7 - Large depth count



Discussion

What is causing this?

Complexities of the fast-paths

	JVSTM	NesTM	PNSTM
read	**O(1)**	$O(1)$	$O(1)$
write	$O(1)$	**O(1)**	$O(1)$

Fast-paths occurrence

	Fast-path	Slow-path
JVSTM	0.99	0.01
NesTM	0.39	0.61
PNSTM	0.39	0.61

Fast-paths occurrence

	Fast-path	Slow-path	Time (μs)
JVSTM	0.99	0.01	1046
NesTM	0.39	0.61	5200
PNSTM	0.39	0.61	7357

Conflicts detected

	Conflicts
JVSTM	845
NesTM	1627
PNSTM	84496

Conflict detection

	JVSTM	NesTM	PNSTM
r-r	-	-	yes
r-w	yes	yes	yes
w-w	yes (if nested)	yes	yes

Conflict detection

	JVSTM	NesTM	PNSTM
r-r	-	-	yes
r-w	yes	yes	yes
w-w	yes (if nested)	yes	yes

Cheaper complexity bounds, more conflicts detected?

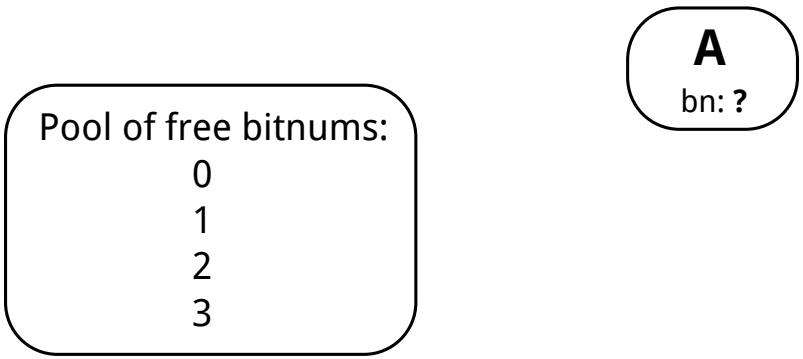
Summary

- Parallel nesting design is coupled with baseline TM
- Complexity analysis may be deceiving
- Average case and conflict detection

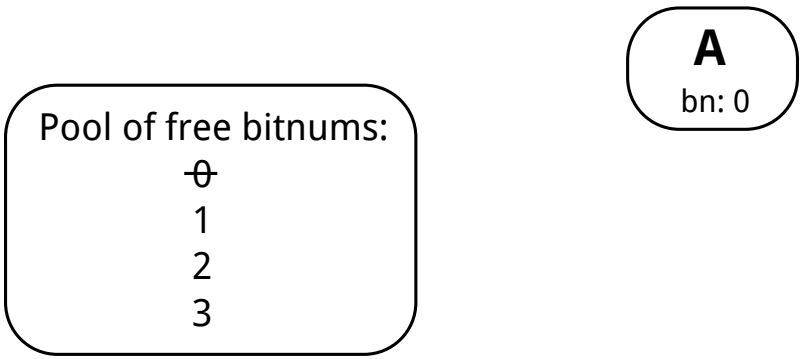
Thank you

Questions?

PNSTM



PNSTM

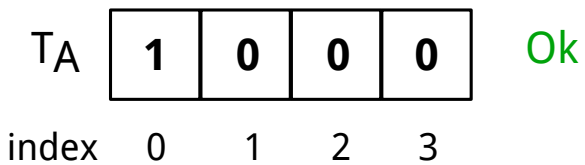


PNSTM

Pool of free bitnums:
~~0~~
1
2
3

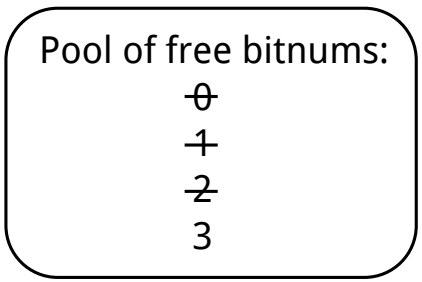
A
bn: 0

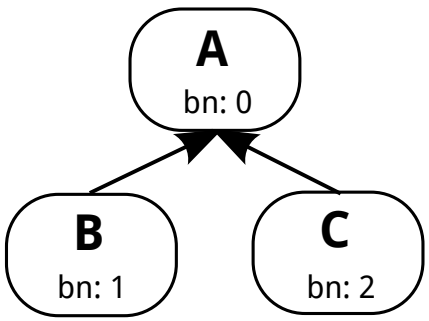
Access Stack of variable X



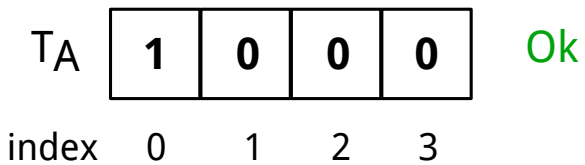
T_A reads X

PNSTM



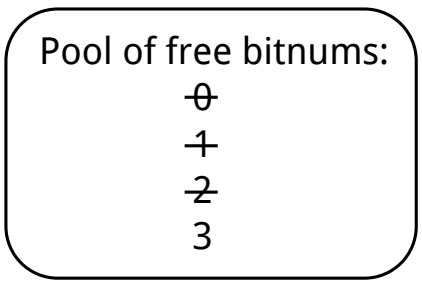


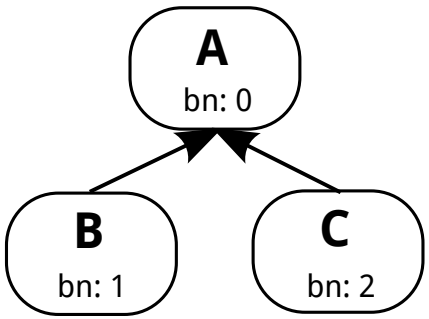
Access Stack of variable X



T_A spawns two children

PNSTM



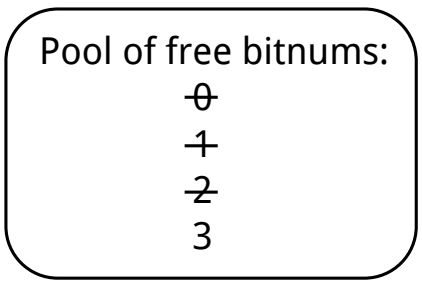


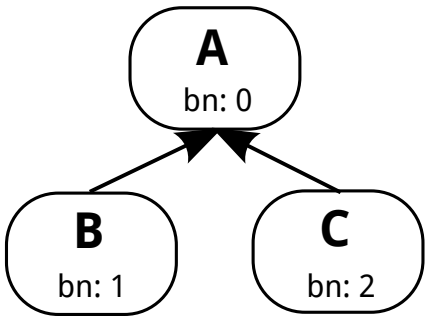
Access Stack of variable X

T_B	**1**	**1**	**0**	**0**	**?**
T_A	**1**	**0**	**0**	**0**	Ok
index	0	1	2	3	

T_B reads X

PNSTM



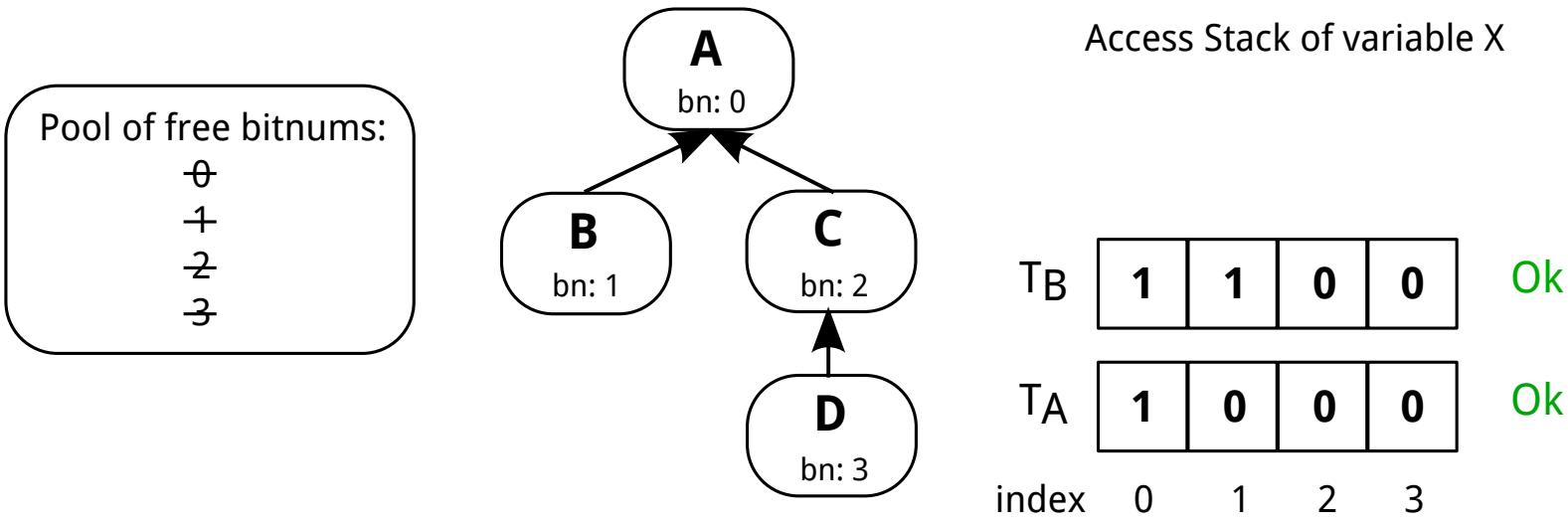


Access Stack of variable X

T_B	**1**	**1**	**0**	**0**	Ok
T_A	**1**	**0**	**0**	**0**	Ok
index	0	1	2	3	

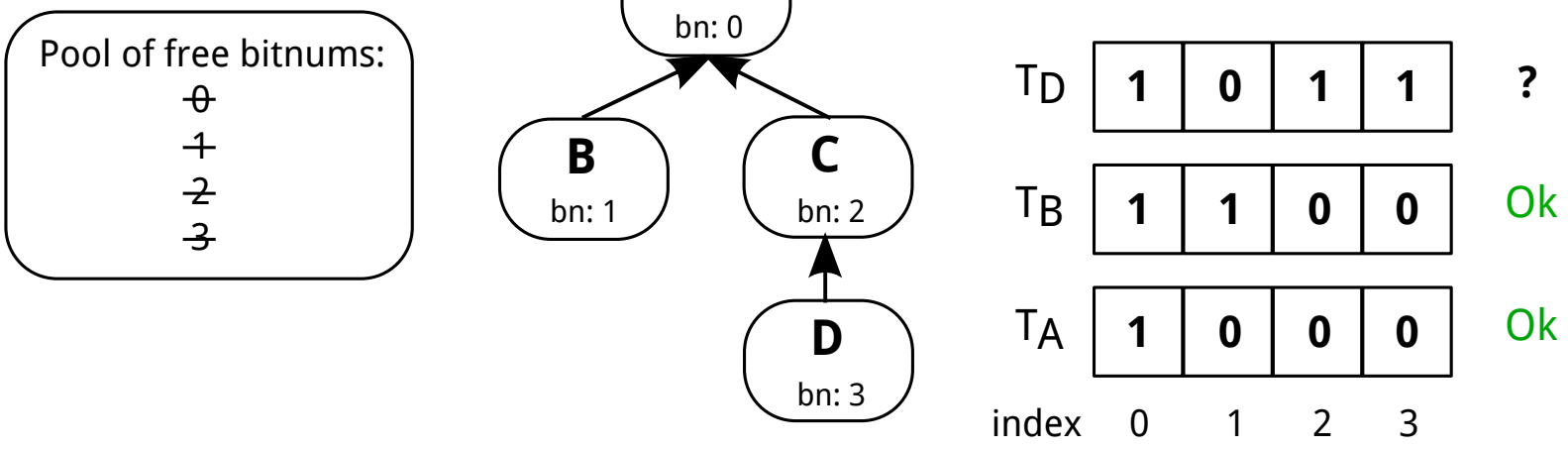
T_B reads X

PNSTM



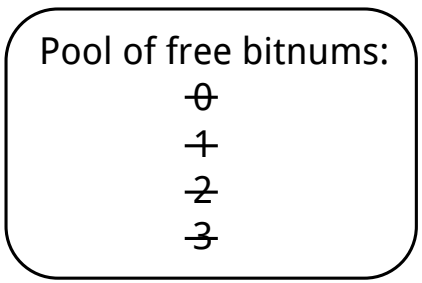
T_B spawns a child

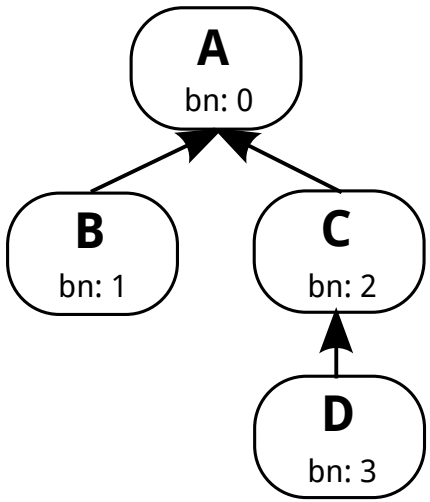
PNSTM

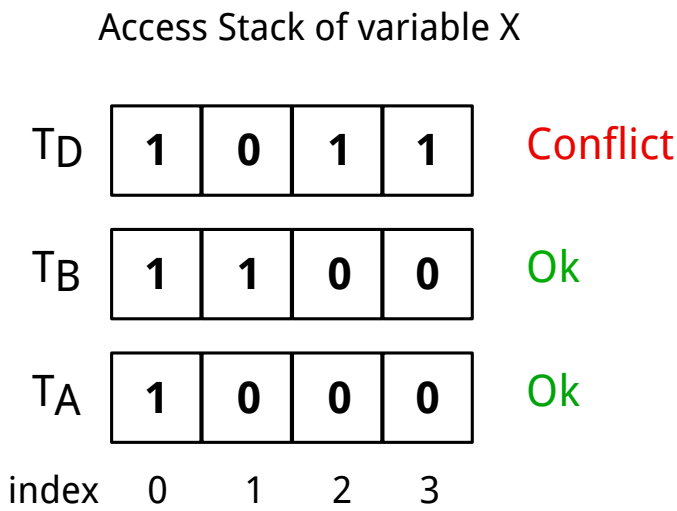


T_D reads X

PNSTM

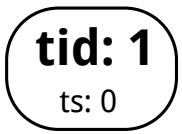






T_D reads X

NesTM

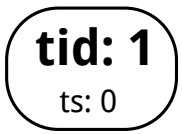


global clock: 0

	timestamp	tid
variable X:	0	0

T_1 starts

NesTM

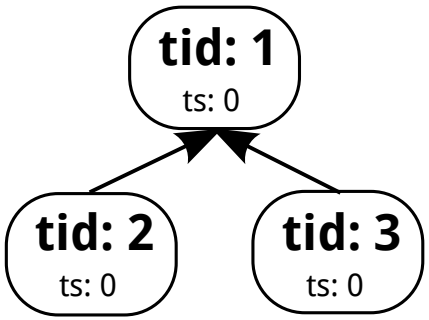


global clock: 0

	timestamp	tid	
variable X:	0	1	Ok

T_1 writes to X

NesTM

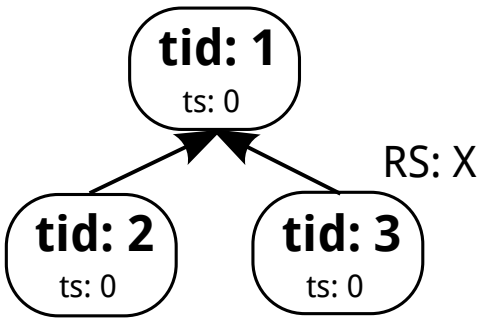


global clock: 0

	timestamp	tid
variable X:	0	1

T_1 spawns two children

NesTM - **read** operation

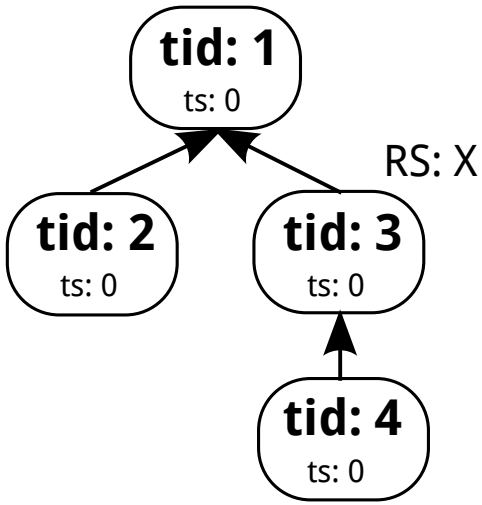


global clock: 0

	timestamp	tid
variable X:	0	1

T_3 reads X

NesTM - **write** operation

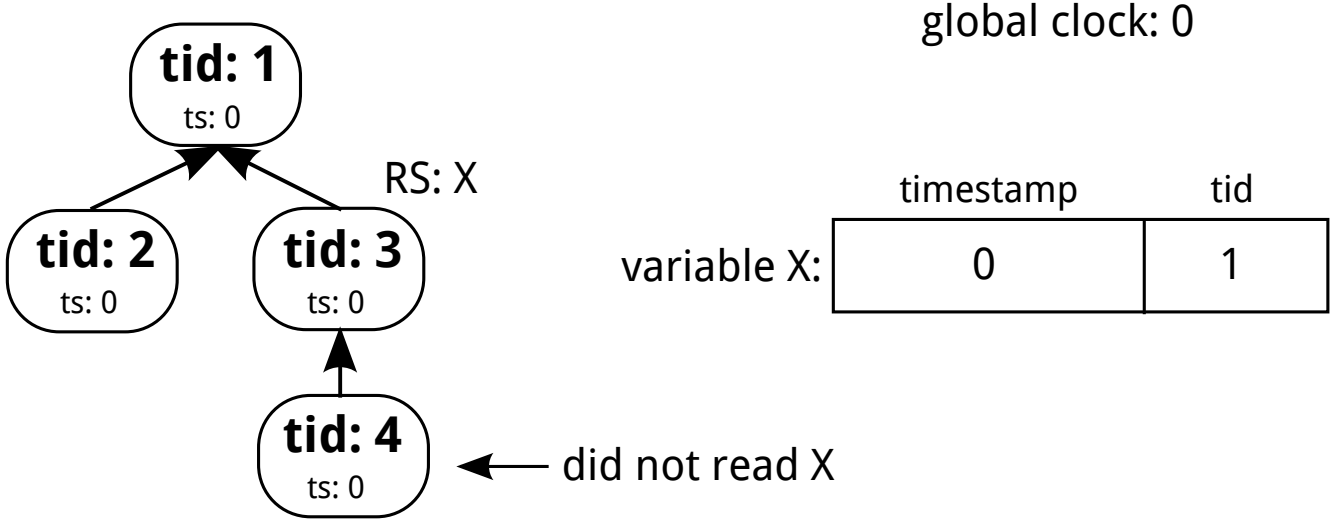


global clock: 0

	timestamp	tid
variable X:	0	1

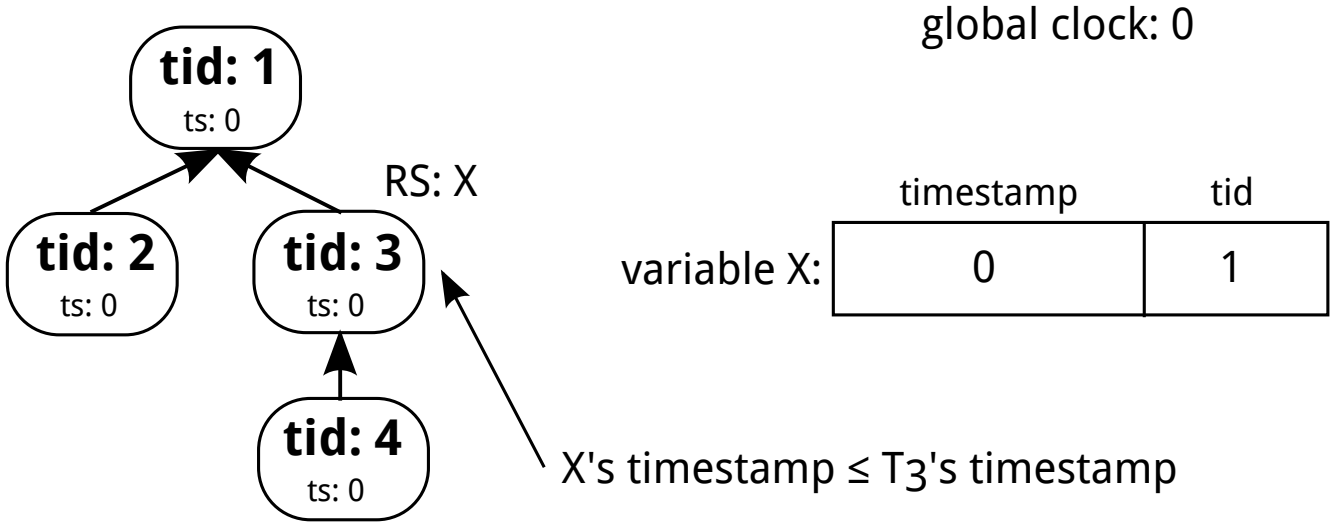
T_3 spawns a child

NesTM - **write** operation



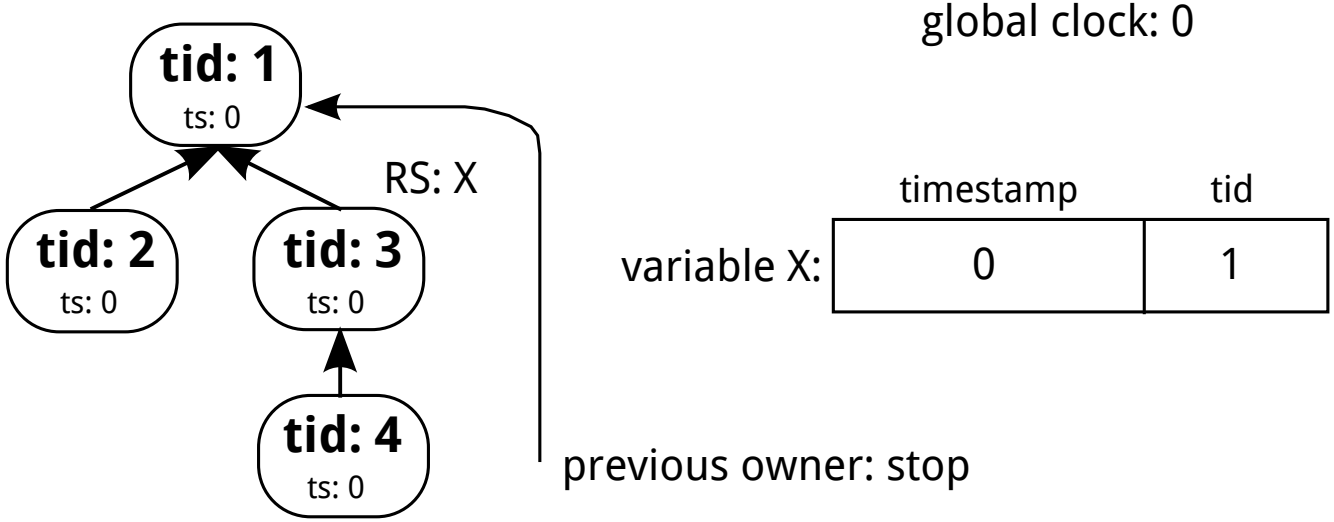
T_4 writes to X

NesTM - **write** operation



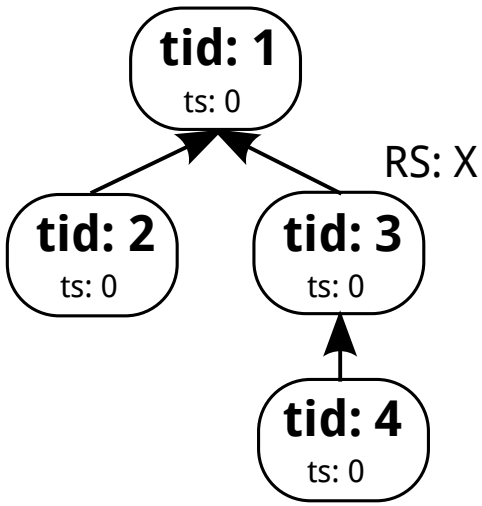
T_4 writes to X

NesTM - **write** operation



T_4 writes to X

NesTM - **write** operation

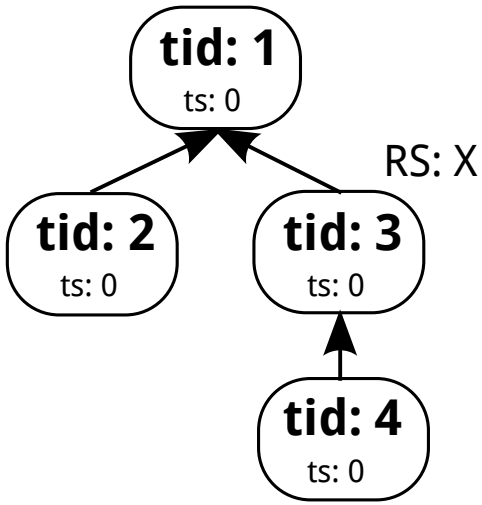


global clock: 0

	timestamp	tid	
variable X:	0	4	Ok

T_4 writes to X

NesTM - **commit** operation

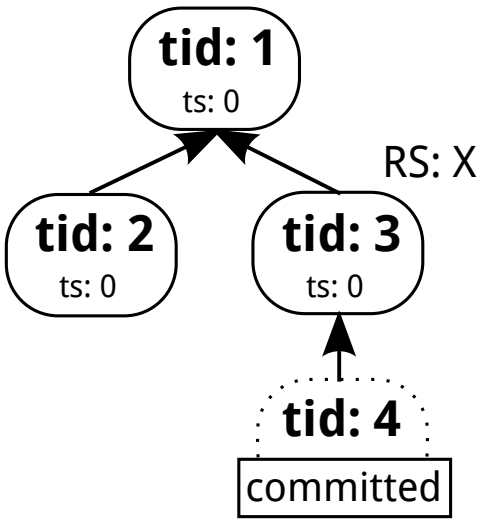


global clock: 0

	timestamp	tid
variable X:	0	4

T_4 prepares commits

NesTM - **commit** operation

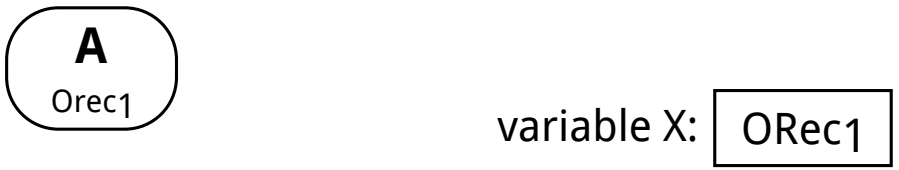


global clock: 1

	timestamp	tid
variable X:	1	3

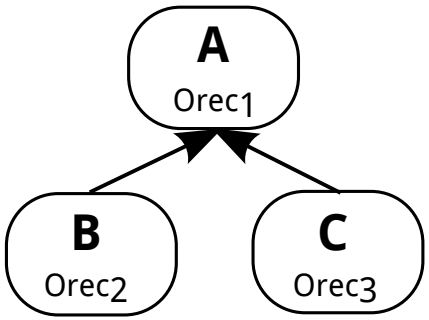
T_4 commits

JVSTM - **write** operation



T_A writes to X

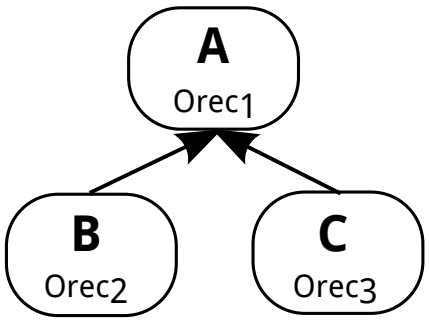
JVSTM - **write** operation

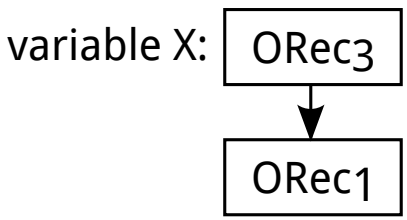


variable X: ORec1

T_A spawns two children

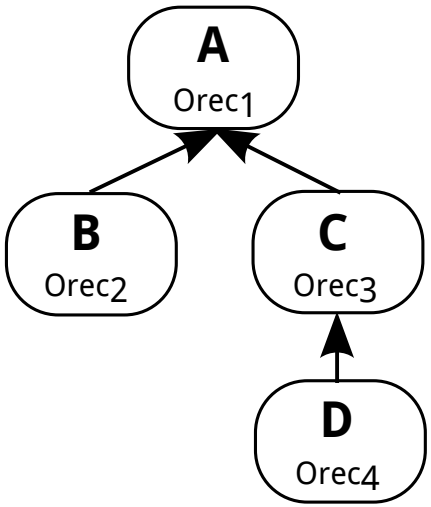
JVSTM - **write** operation

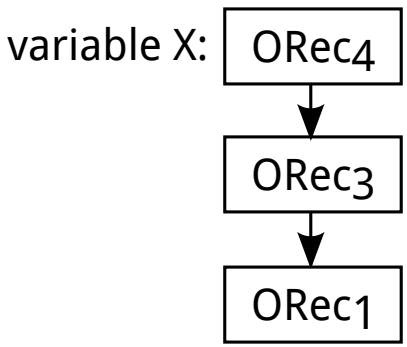




T_C writes to X

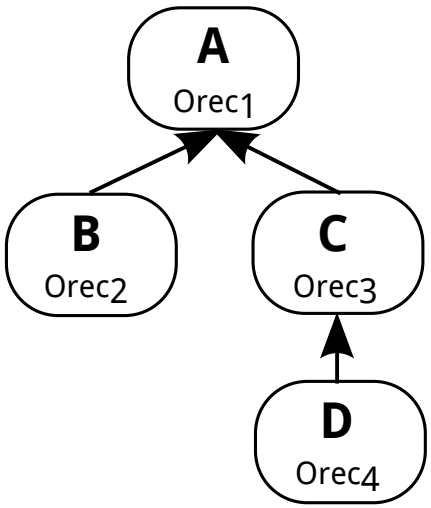
JVSTM - **write** operation

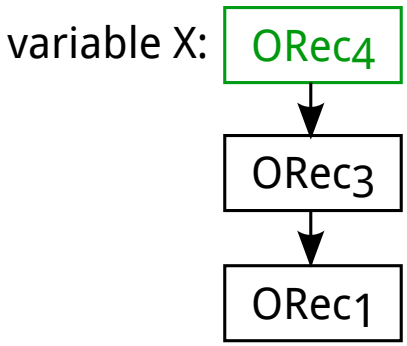




T_D is spawned and writes to X

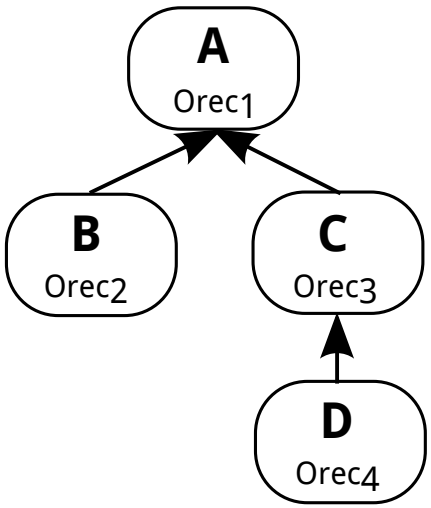
JVSTM - **read** operation

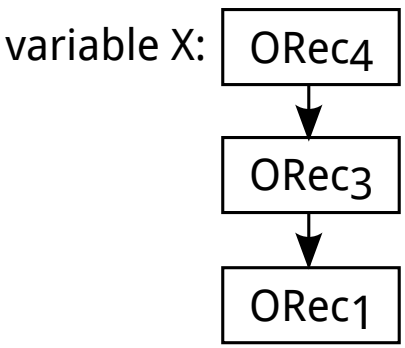




T_D reads X

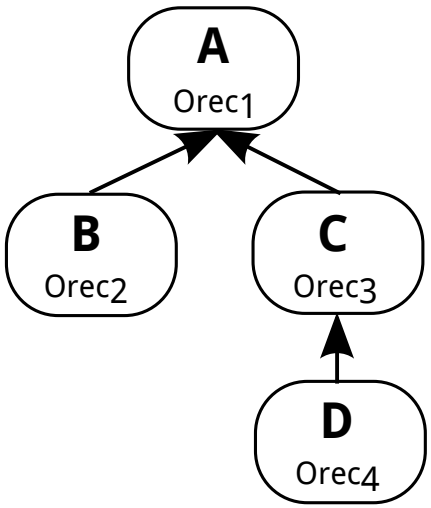
JVSTM - **read** operation

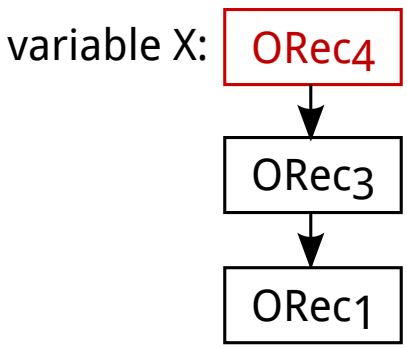




T_B reads X

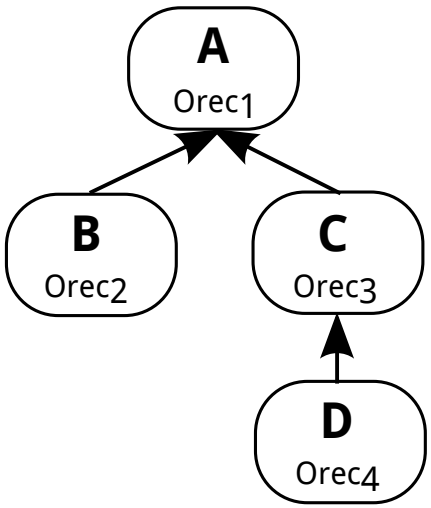
JVSTM - **read** operation

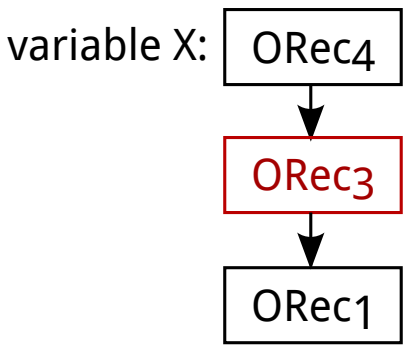




T_B reads X

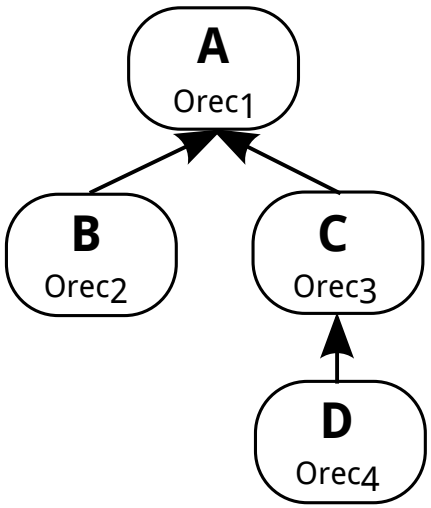
JVSTM - **read** operation

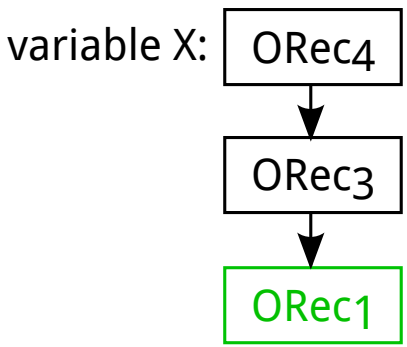




T_B reads X

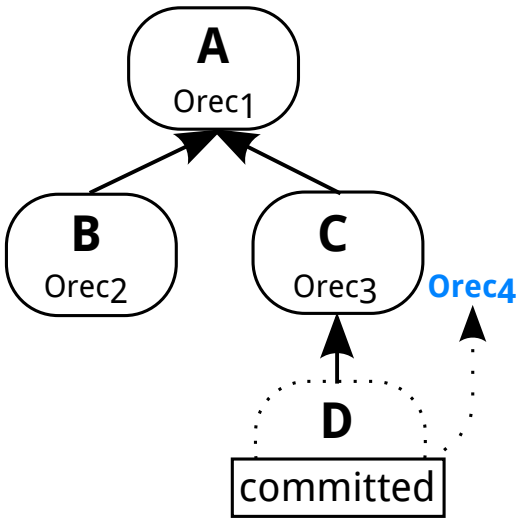
JVSTM - **read** operation

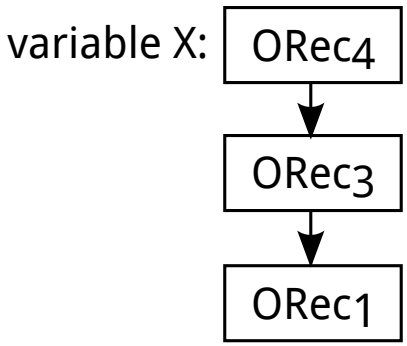




T_B reads X

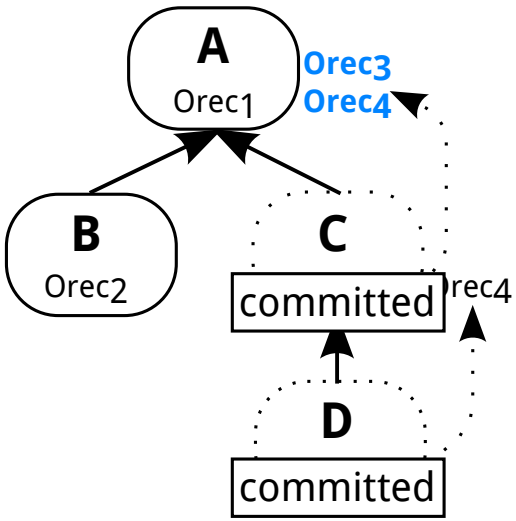
JVSTM - **commit** operation

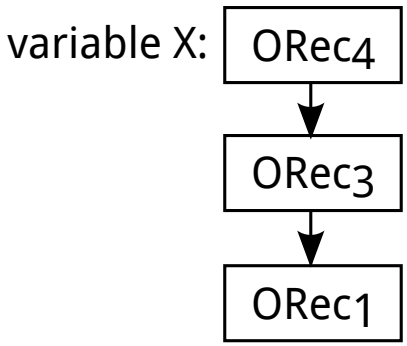




T_D commits

JVSTM - **commit** operation





T_C commits

Evaluation - Top-level txs only