The International Conference on Networked Systems (NETYS-2018)

Efficient means of Achieving Composability using Object based Conflicts on Transactional Memory

Sathya Peri, Ajay Singh and **Archit Somani**
(sathya_p, cs15mtech01001, cs15resch01001)@iith.ac.in
CSE Department, IIT Hyderabad, India

November 1, 2018

Content

Content

Transaction

Sequence of instructions guaranteed to execute atomically.

Transaction

Sequence of instructions guaranteed to execute atomically.

History

Concurrent execution of transactions.

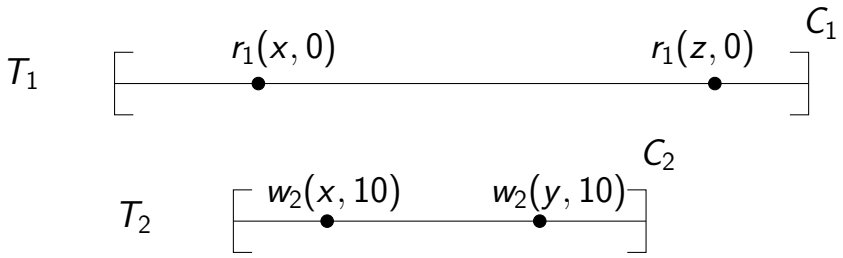

Figure: History of an STM

STMs Contd..
Introduction

- **Software Transaction Memory Systems(STMs)** are a convenient programming interface for a programmer to access shared memory using concurrent threads without worrying about concurrency issues.

STMs Contd..

Introduction

- **Software Transaction Memory Systems(STMs)** are a convenient programming interface for a programmer to access shared memory using concurrent threads without worrying about concurrency issues.
- STMs export the following methods:
  - t_begin(),
  - t_read(),
  - t_write(),
  - tryC() and tryA().

  We refer to these as Read-Write STMs(or RWSTM).

STMs Contd..

Working of STM System

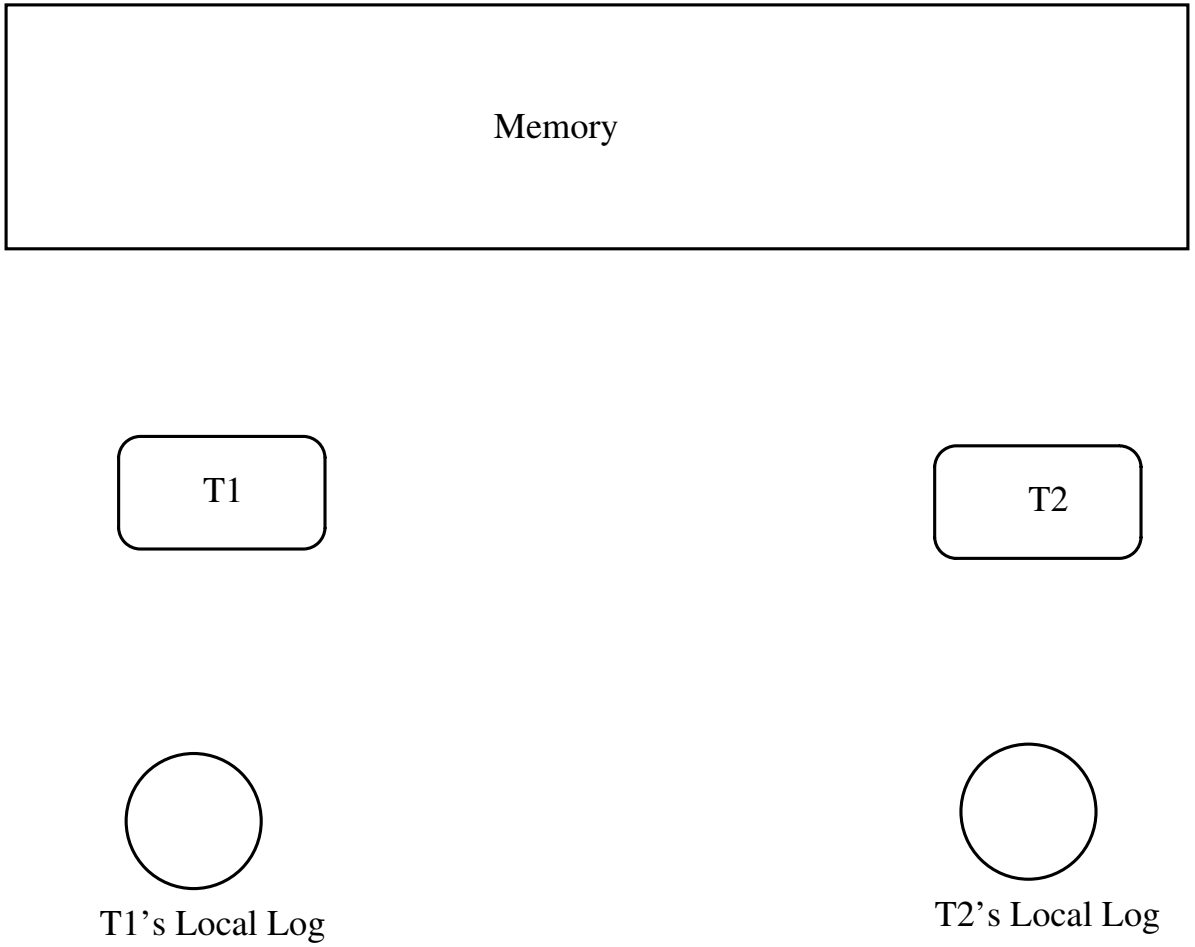

Figure: Working of STM System

STMs Contd..

Working of STM System

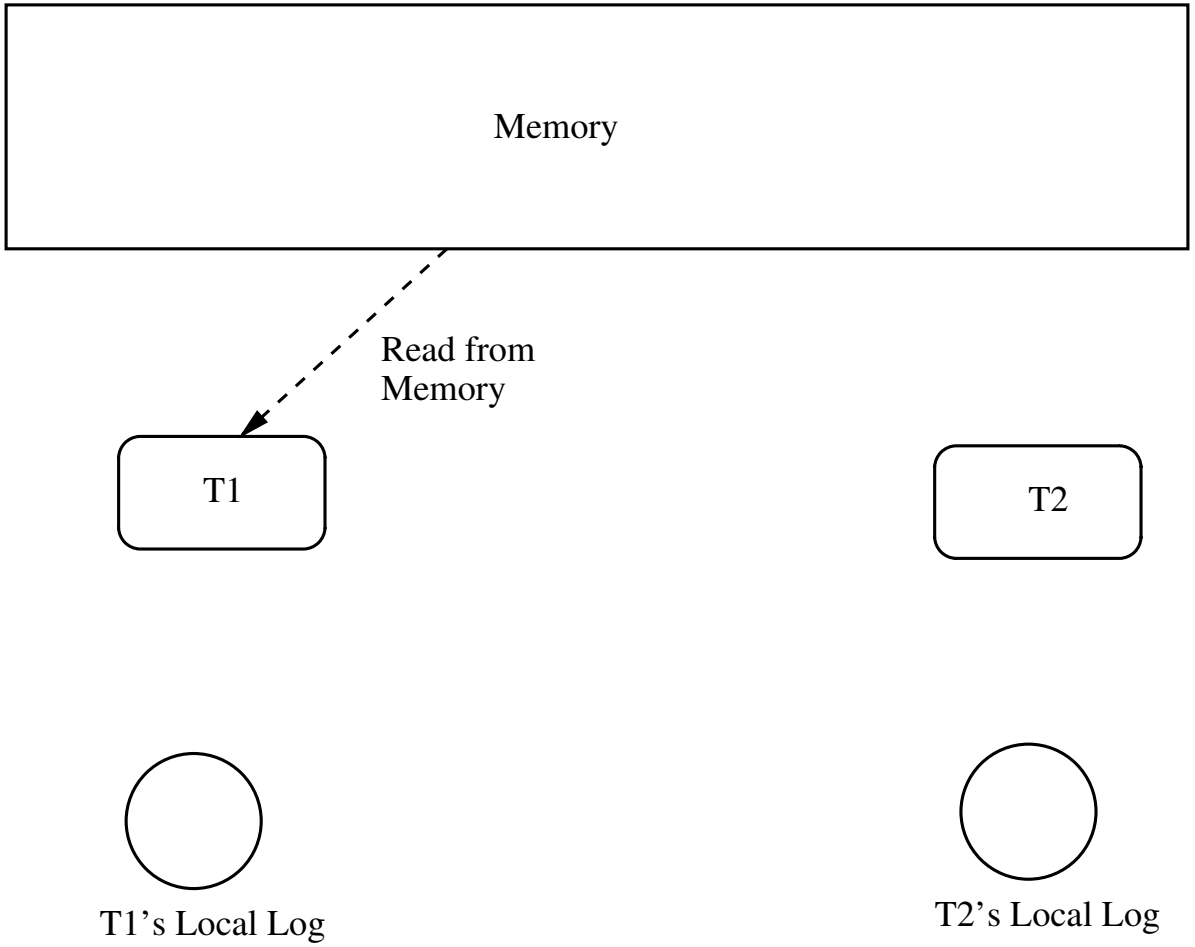

Figure: Working of STM System

STMs Contd..

Working of STM System

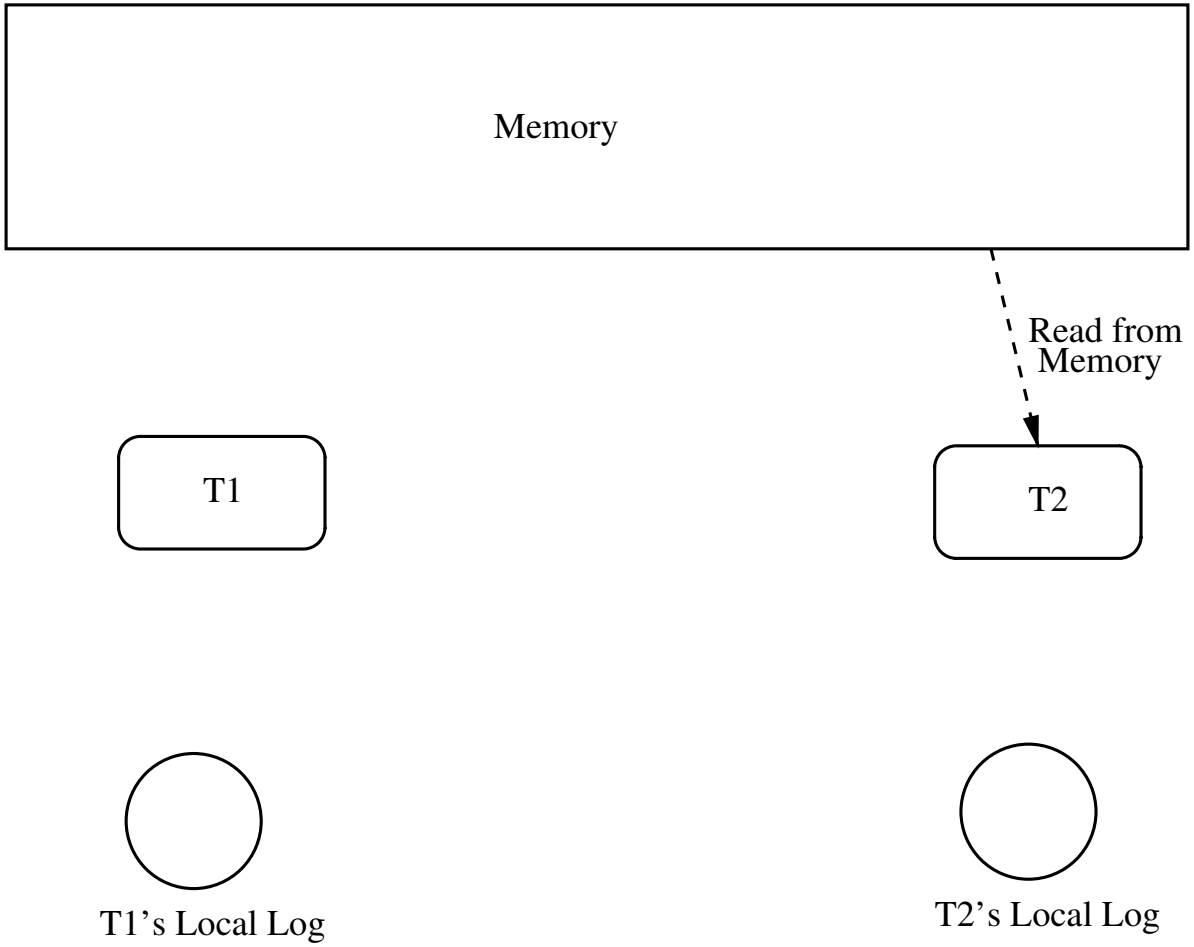

Figure: Working of STM System

STMs Contd..

Working of STM System

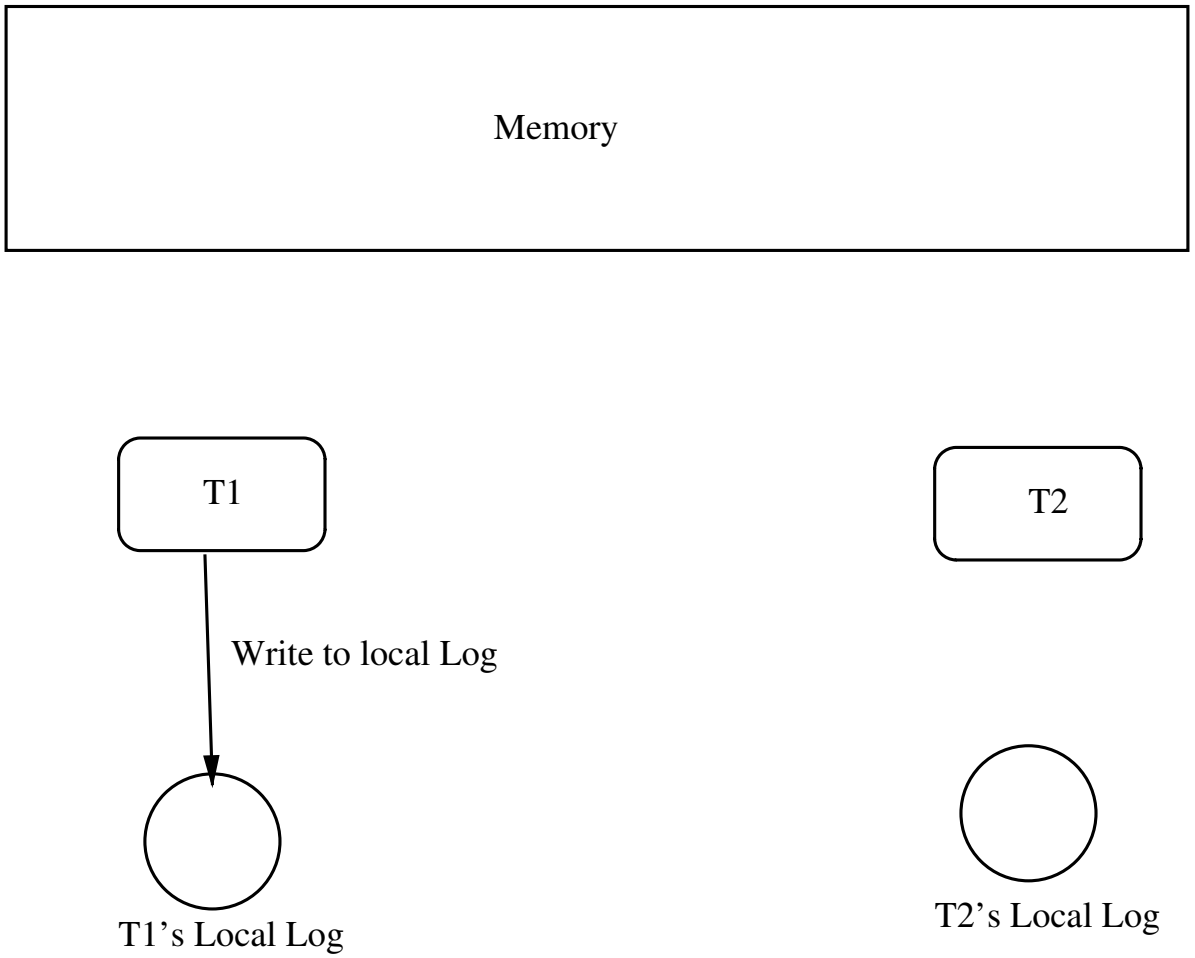

Figure: Working of STM System

STMs Contd..

Working of STM System

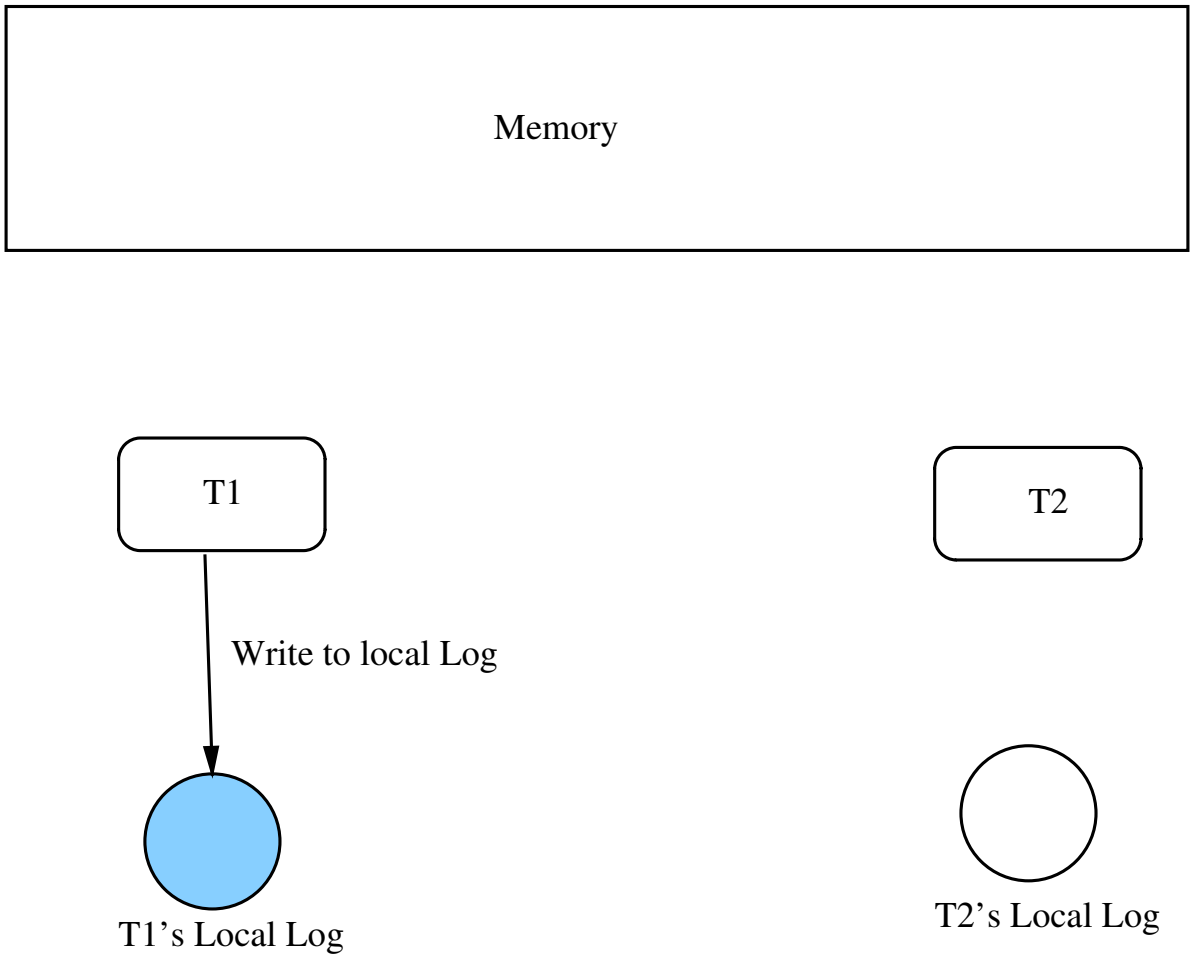

Figure: Working of STM System

STMs Contd..

Working of STM System

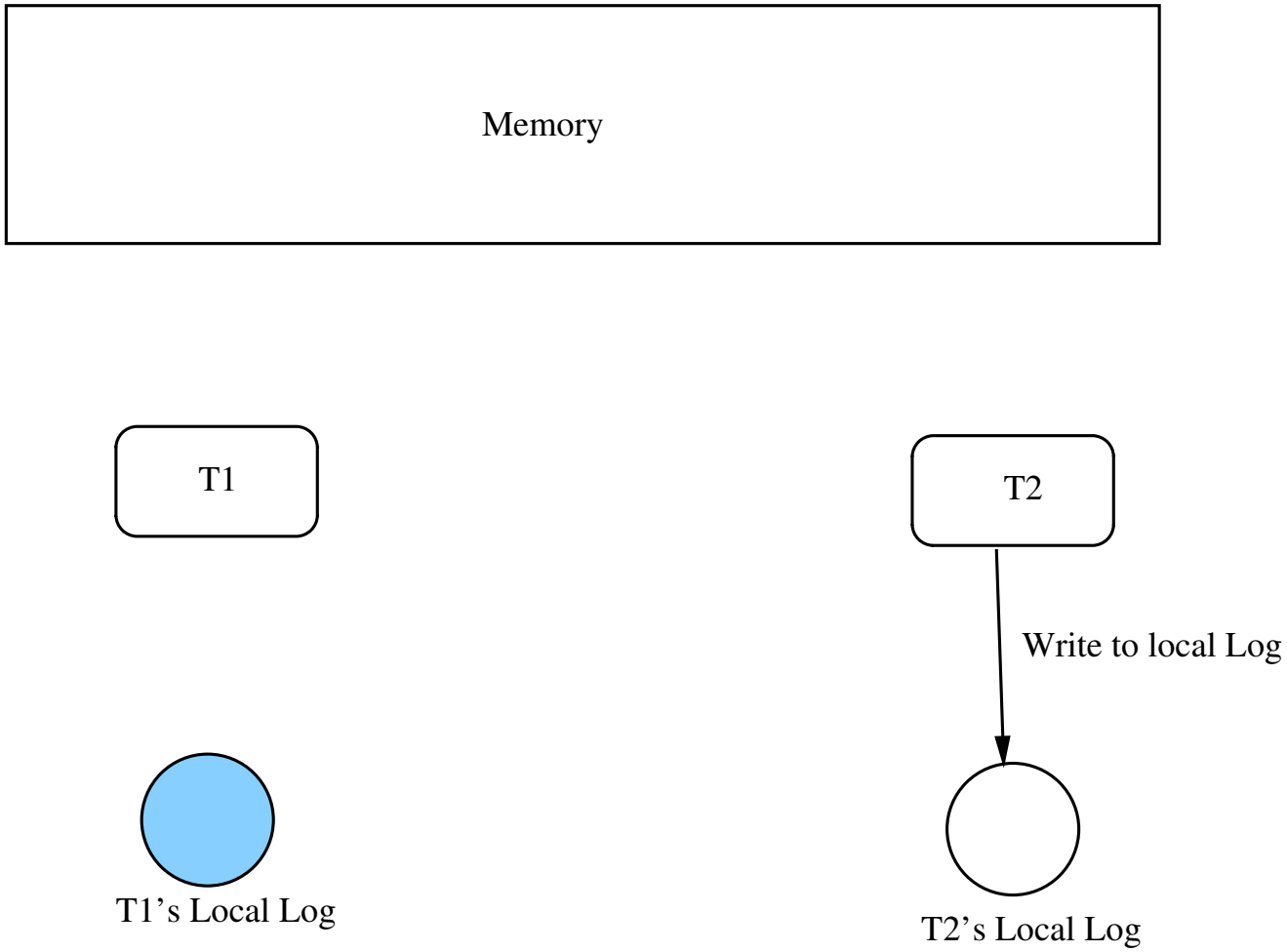

Figure: Working of STM System

STMs Contd..

Working of STM System

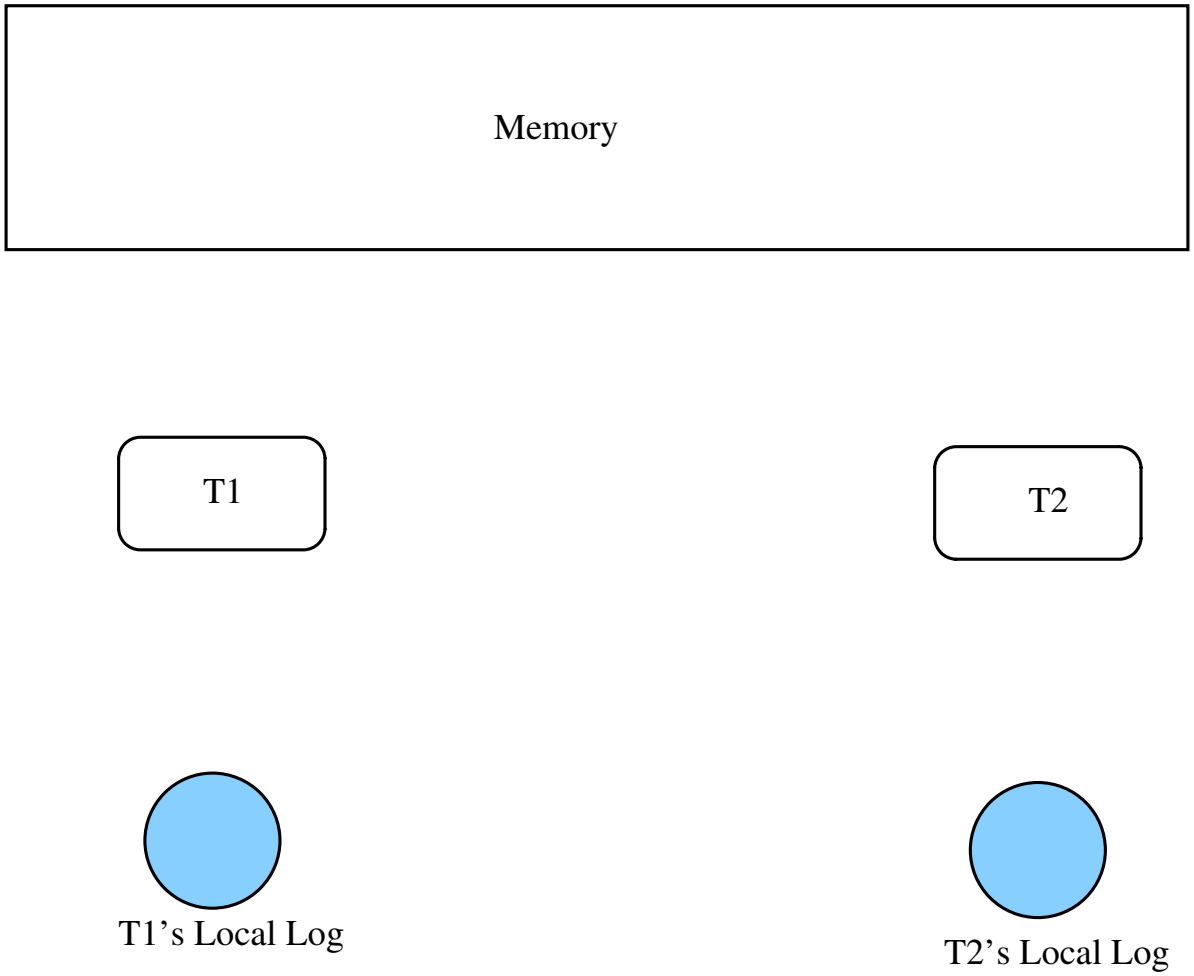

Figure: Working of STM System

STMs Contd..
Working of STM System

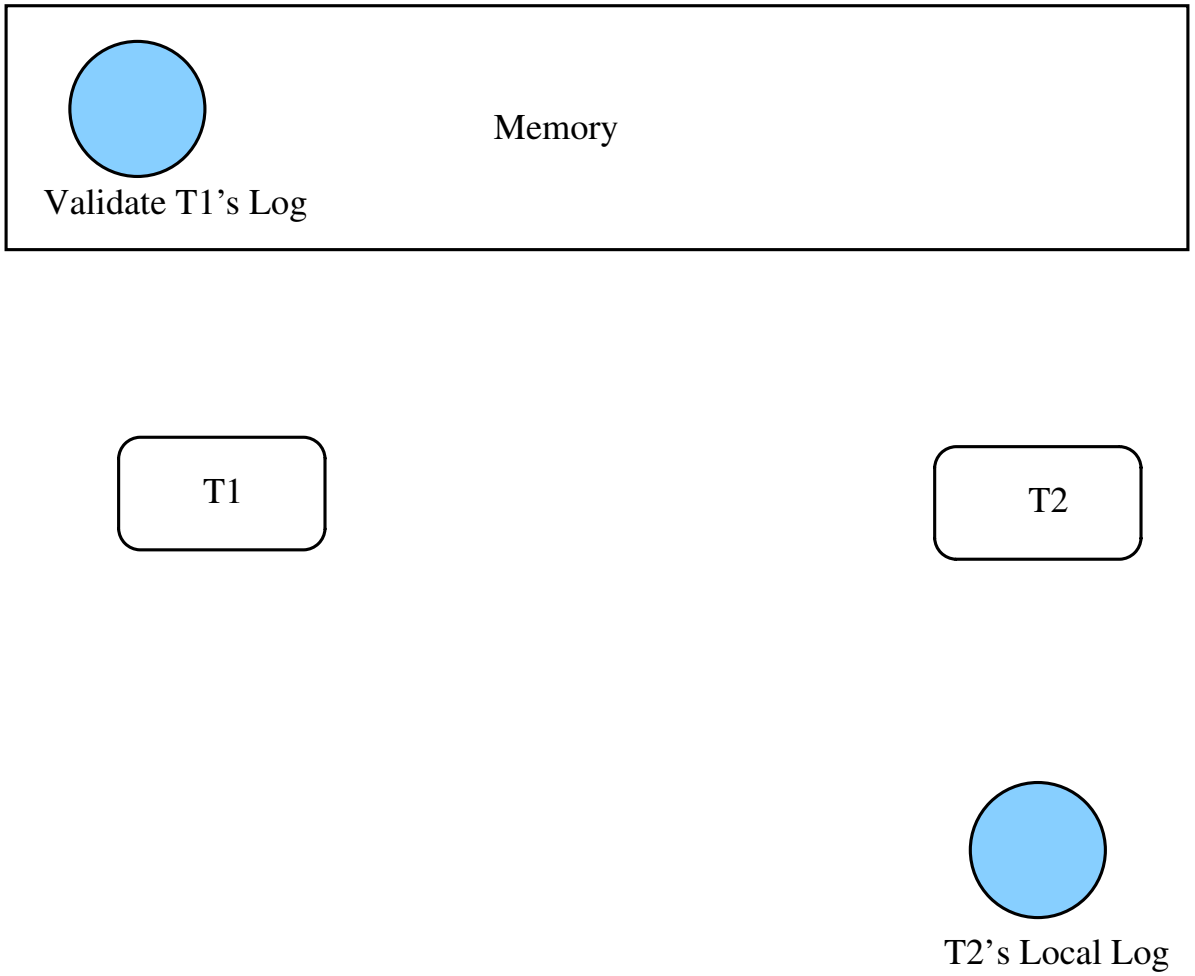

Figure: Working of STM System

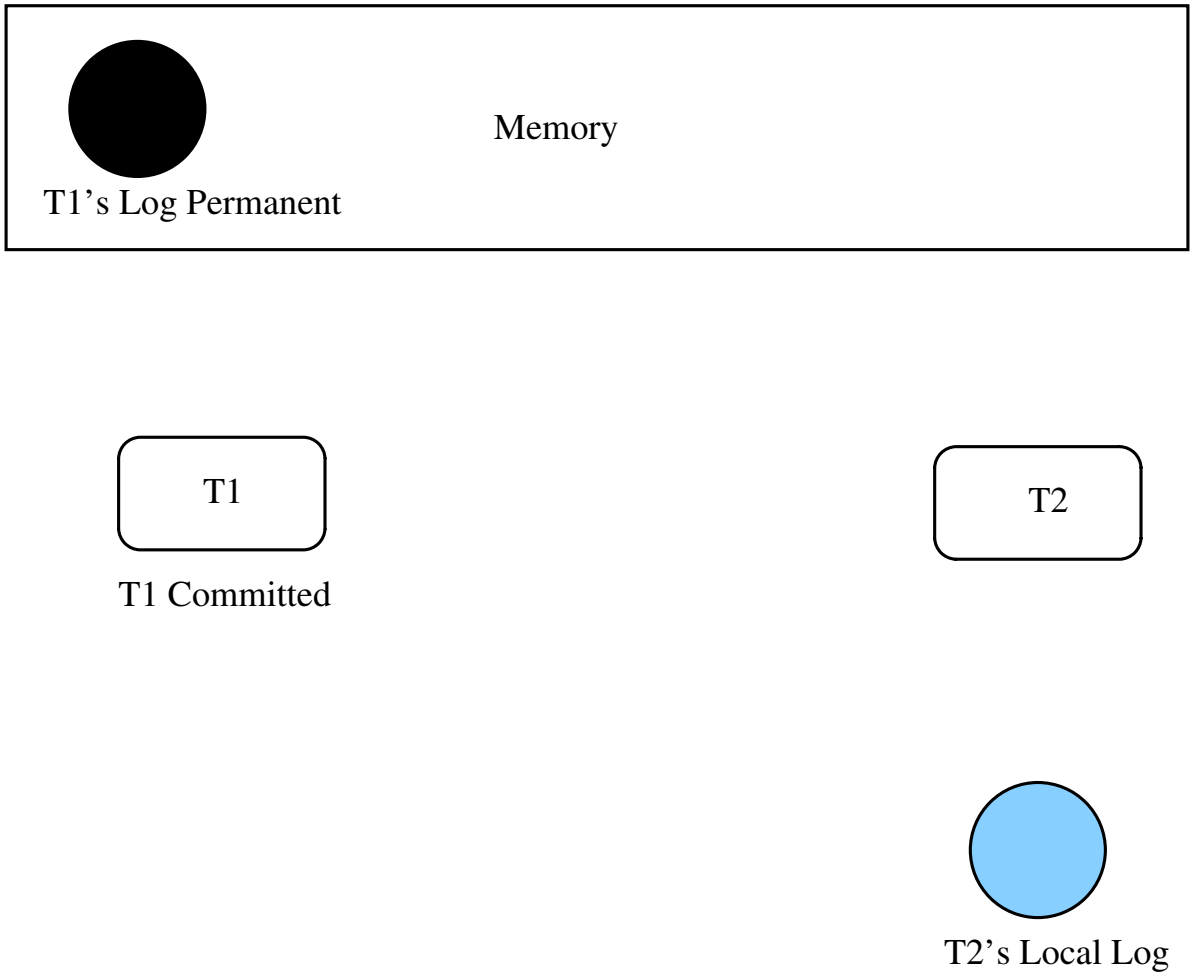

Figure: Working of STM System

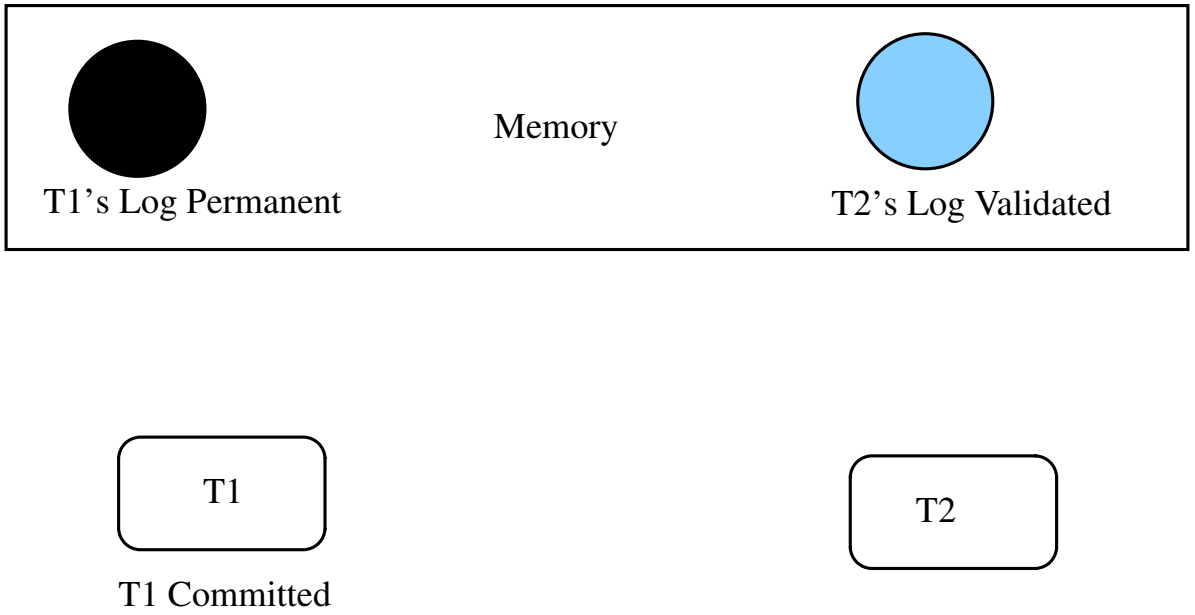

Figure: Working of STM System

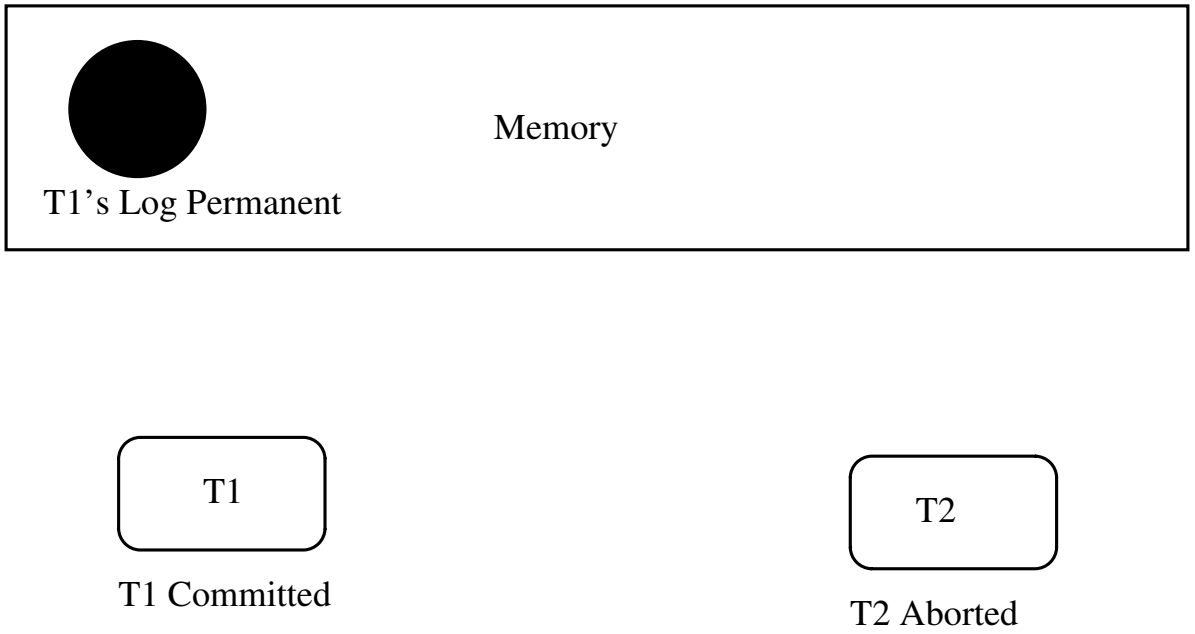

Figure: Working of STM System

Correctness Criterion: Opacity *[Guerraoui and Kapałka]*

- A history H is opaque if there exists a serial history S s.t.
  1. Operations of H and S are same
  2. S respects real time order $\prec_H^{RT}$ and
  3. $\forall$ trans(T_i) $\in$ S is legal.

Correctness Criterion: Opacity *[Guerraoui and Kapałka]*

- A history H is opaque if there exists a serial history S s.t.
  1. Operations of H and S are same
  2. S respects real time order $\prec_H^{RT}$ and
  3. $\forall$ trans(T_i) $\in$ S is legal.

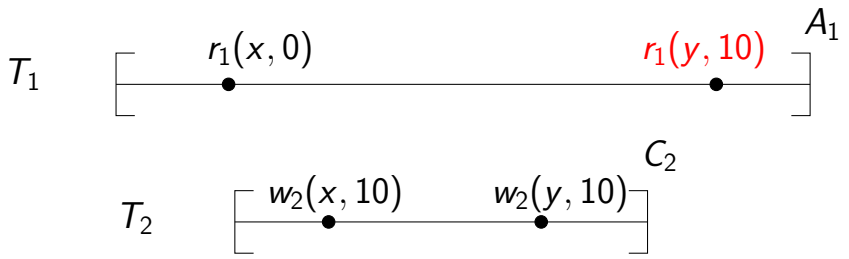

Figure: History H is not Opaque

Correctness Criterion: Opacity *[Guerraoui and Kapałka]*

- A history H is opaque if there exists a serial history S s.t.
  1. Operations of H and S are same
  2. S respects real time order $\prec_H^{RT}$ and
  3. $\forall$ trans(T_i) $\in$ S is legal.

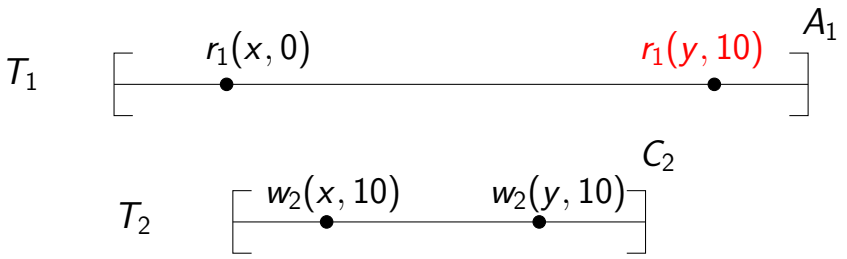

Figure: History H is not Opaque

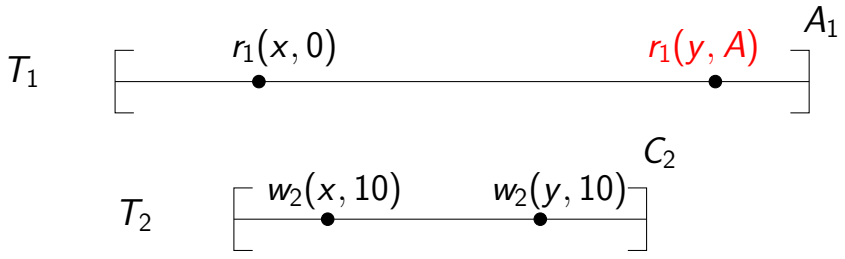

Figure: Opaque History H(T_1, T_2)

Content

Problem with Read-Write STM

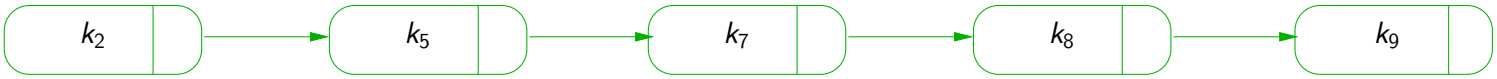

Figure: A concurrent list

Problem with Read-Write STM

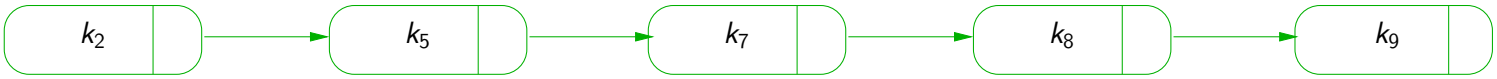

Figure: A concurrent list

T1: lookup(k_5), lookup(k_8) & T_2: delete(k_7)

Problem with Read-Write STM

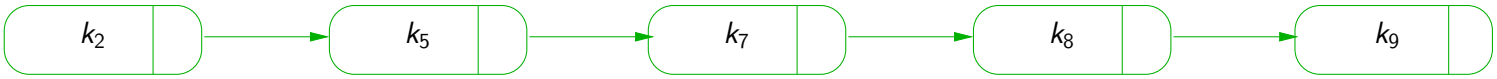

Figure: A concurrent list

T1: lookup(k_5), lookup(k_8) & T_2: delete(k_7)

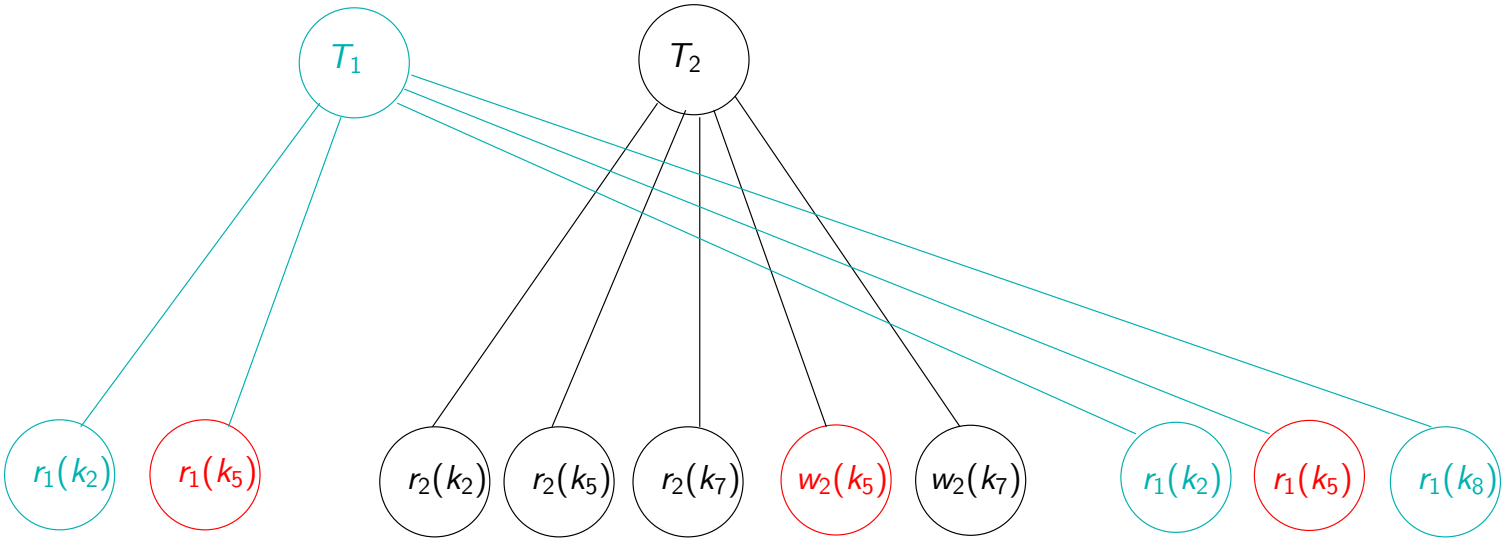

Figure: Transaction Tree Structure

Problem with Read-Write STM

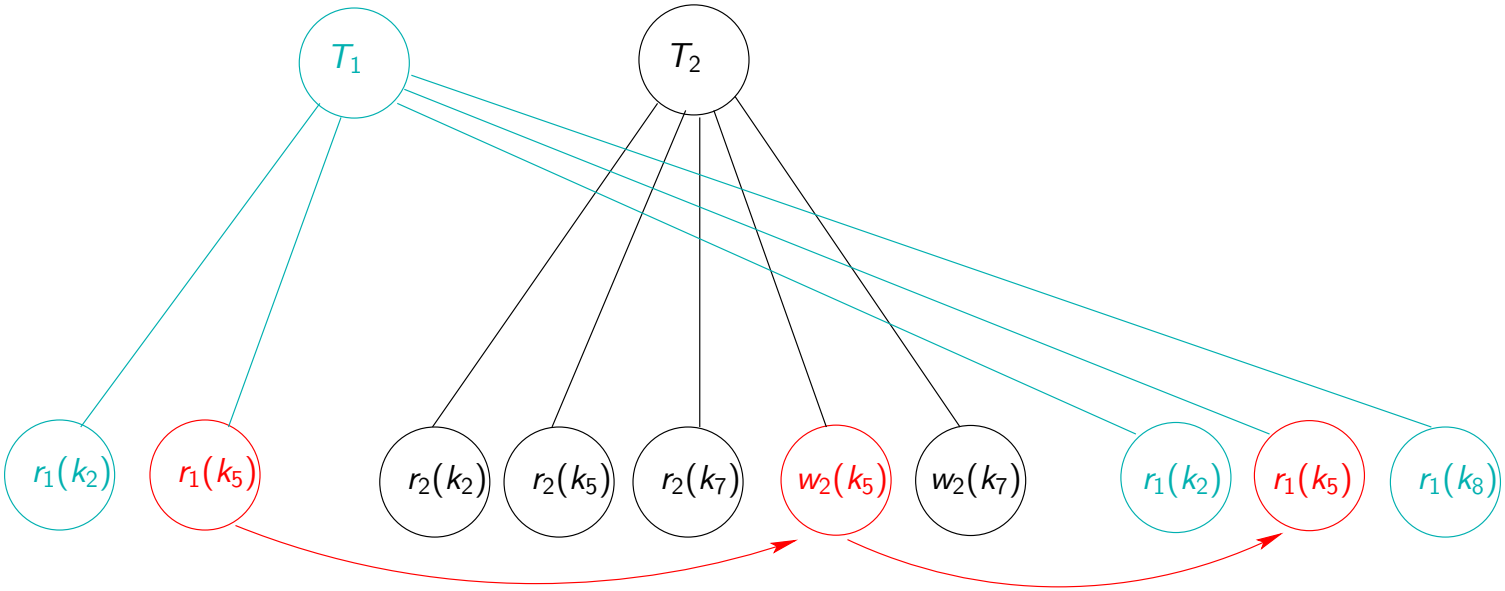

Figure: Cyclic Conflicts

Schedule could not be accepted by STM.

Problem with Read-Write STM

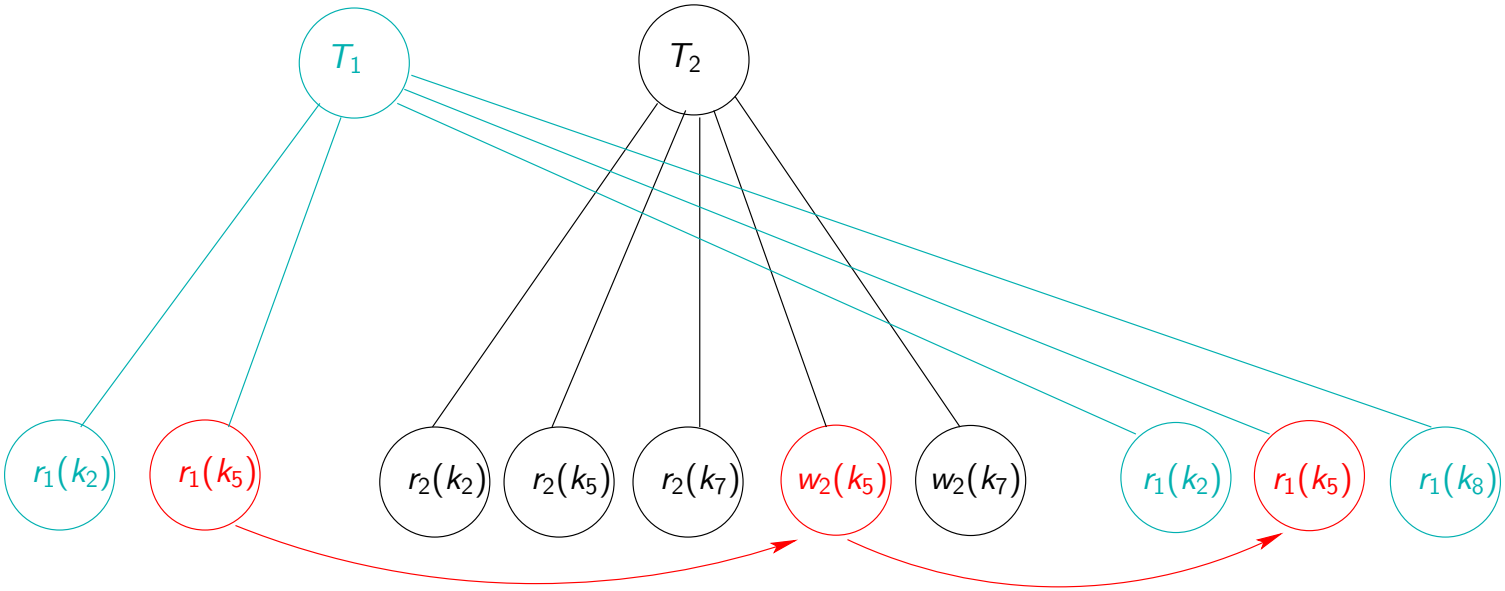

Figure: Cyclic Conflicts

Schedule could not be accepted by STM.

Figure: Cycle (Not Serial)

Content

Motivation towards Object based STM (OSTM)

Opportunity to Leverage CDS semantics

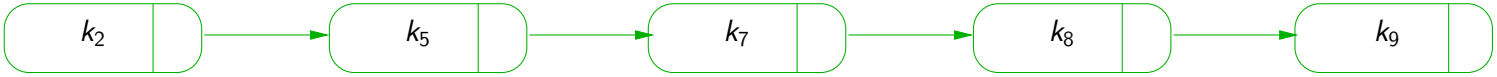

Figure: A concurrent list

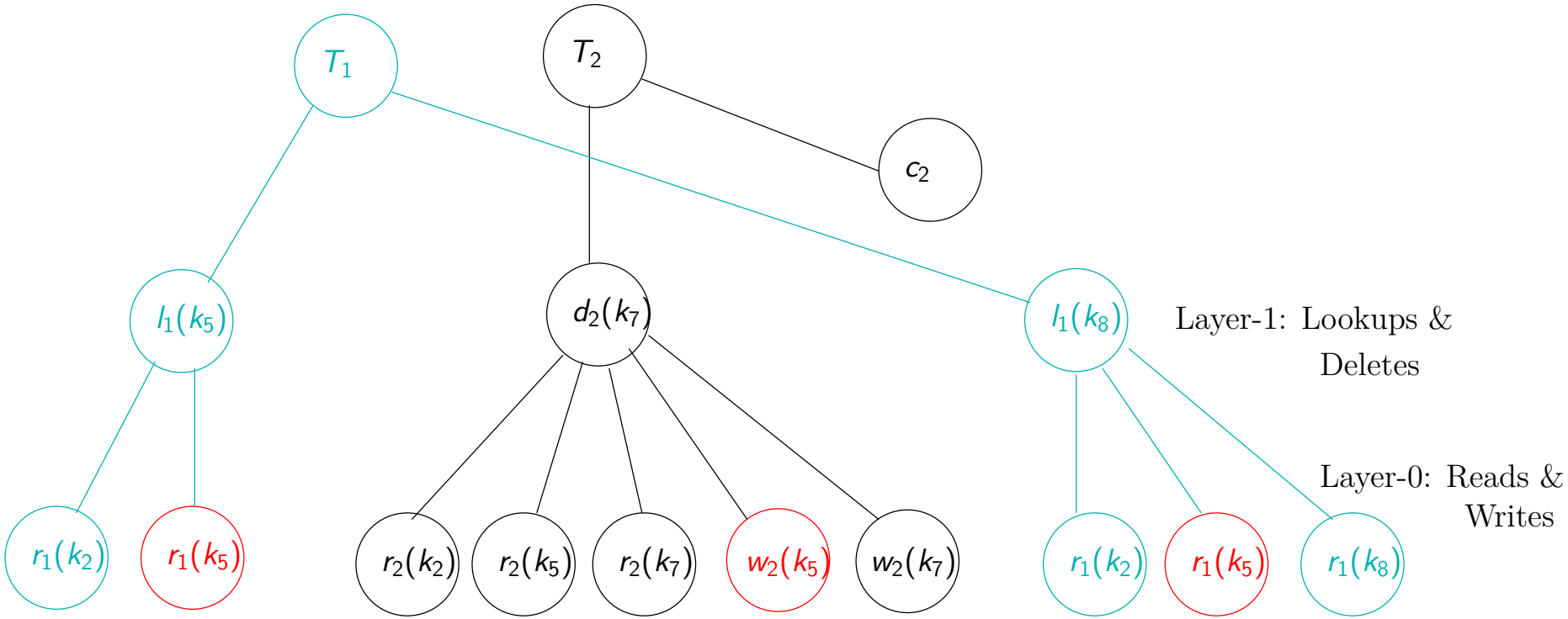

Figure: Transaction Tree Structure *[Weikum and Vossen]*

Level-0 conflicts are irrelevant at Level-1.

Motivation towards Object based STM (OSTM)

Opportunity to Leverage CDS semantics

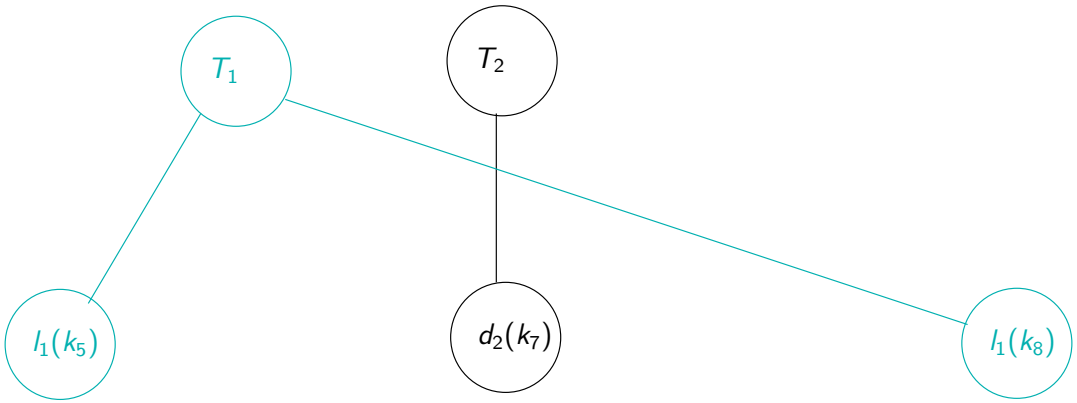

Figure: Pruned Tree

Motivation towards Object based STM (OSTM)

Opportunity to Leverage CDS semantics

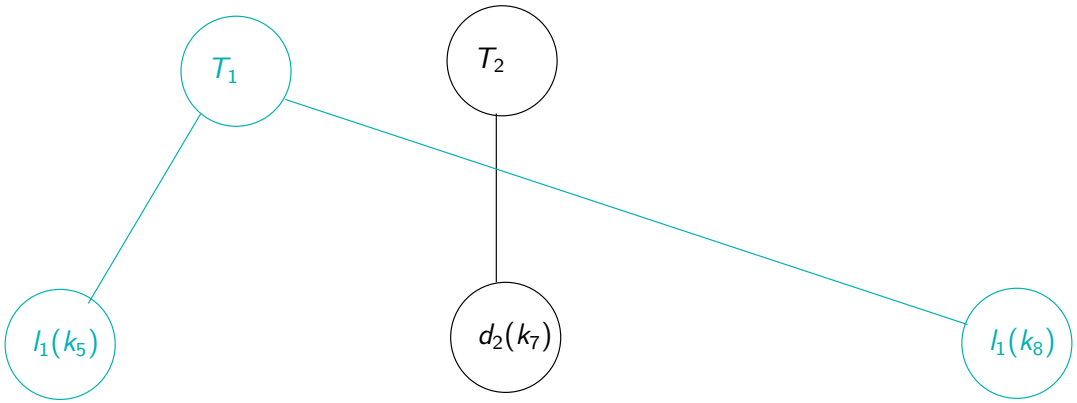

Figure: Pruned Tree

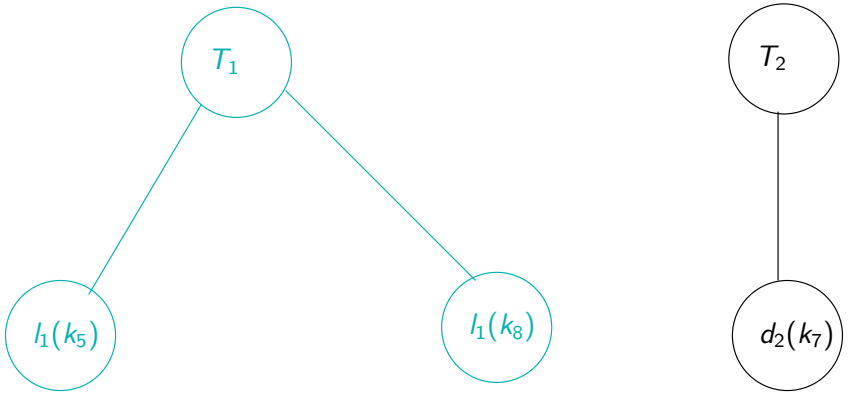

Figure: Sequential Schedule

Motivation towards Object based STM (OSTM)

Opportunity to Leverage CDS semantics

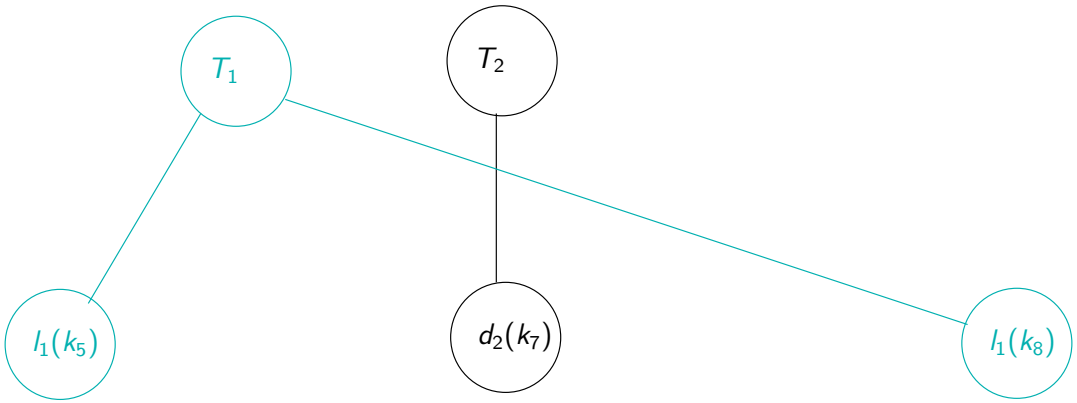

Figure: Pruned Tree

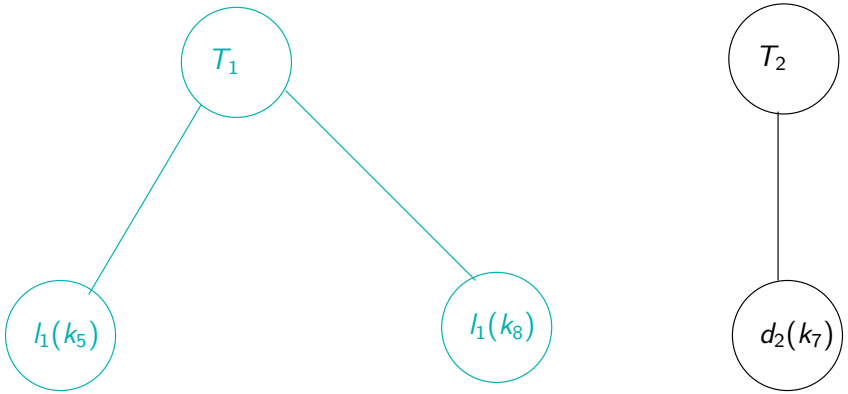

Figure: Sequential Schedule

Figure: Serial History

Schedule can be accepted by STM.

Million Dollar Question

Can we gain more concurrency yet ensuring composability and ease of programming?

Million Dollar Question

Can we gain more concurrency yet ensuring composability and ease of programming?

Solution: Yes

Million Dollar Question

Can we gain more concurrency yet ensuring composability and ease of programming?

Solution: Yes

- Ease of programming ⇒ STM Interface.
- Efficient Composition ⇒ Object Level Semantics.

Proposed OSTM Model
Introduction

- **Object-based STMs (OSTM)** operate on higher level objects rather than primitive reads & writes which act upon memory locations.

Proposed OSTM Model

Introduction

- **Object-based STMs (OSTM)** operate on higher level objects rather than primitive reads & writes which act upon memory locations.
- OSTM model exports:
  - *STM_begin()*, *STM_push()*, *STM_pop()*, *STM_peek()* and *STM_tryC()* for stack.
  - *STM_begin()*, *STM_insert()*, *STM_delete()*, *STM_lookup() and STM_tryC()* for set.

OSTM: Detail View

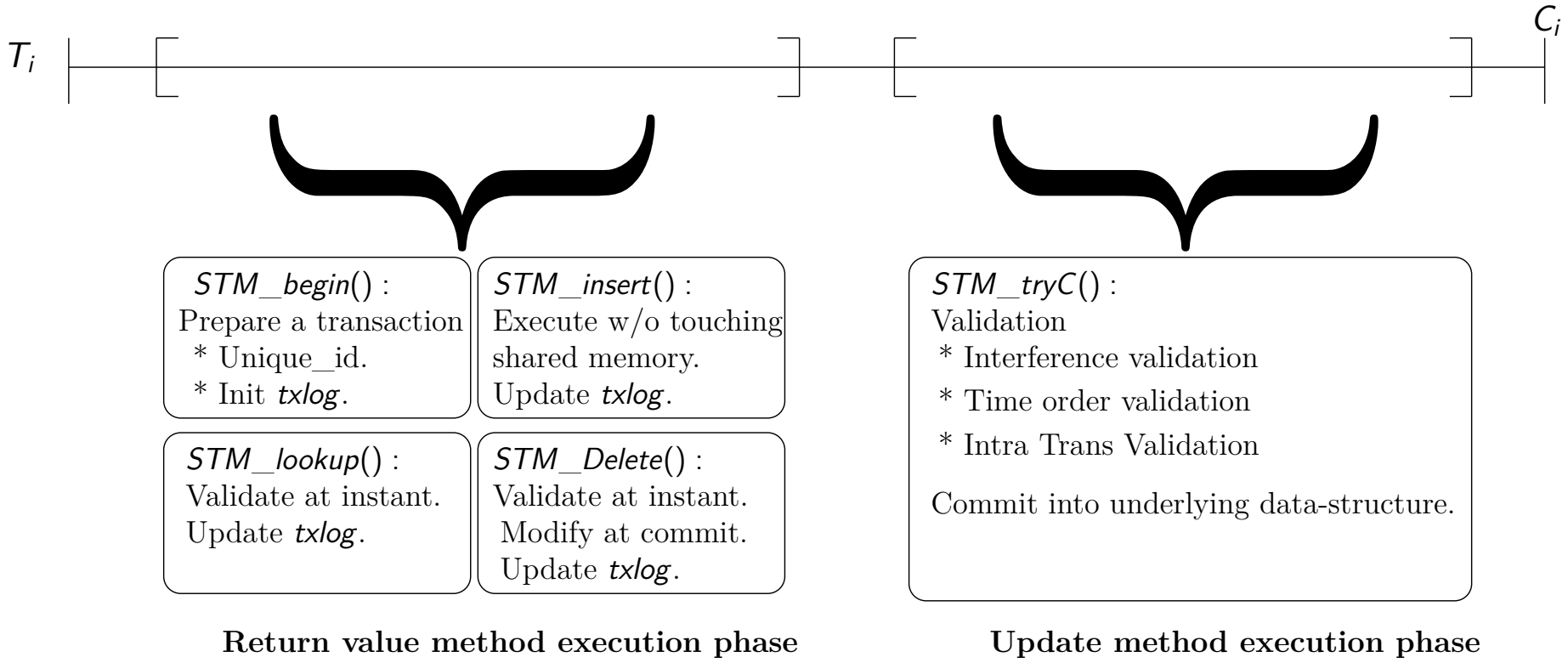

Figure: Transaction lifecycle of OSTM

Content

HT-OSTM: Design
Underlying Data-Structure

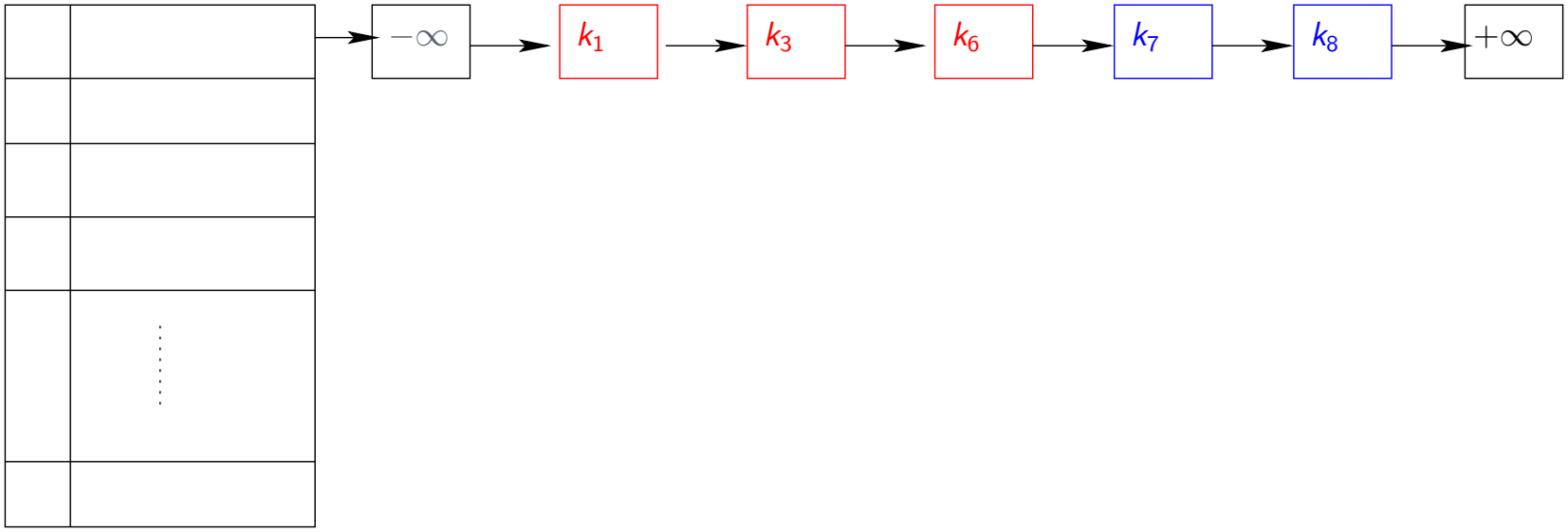

Figure: The underlying shared data-structure as hash table

HT-OSTM: Design

Underlying Data-Structure

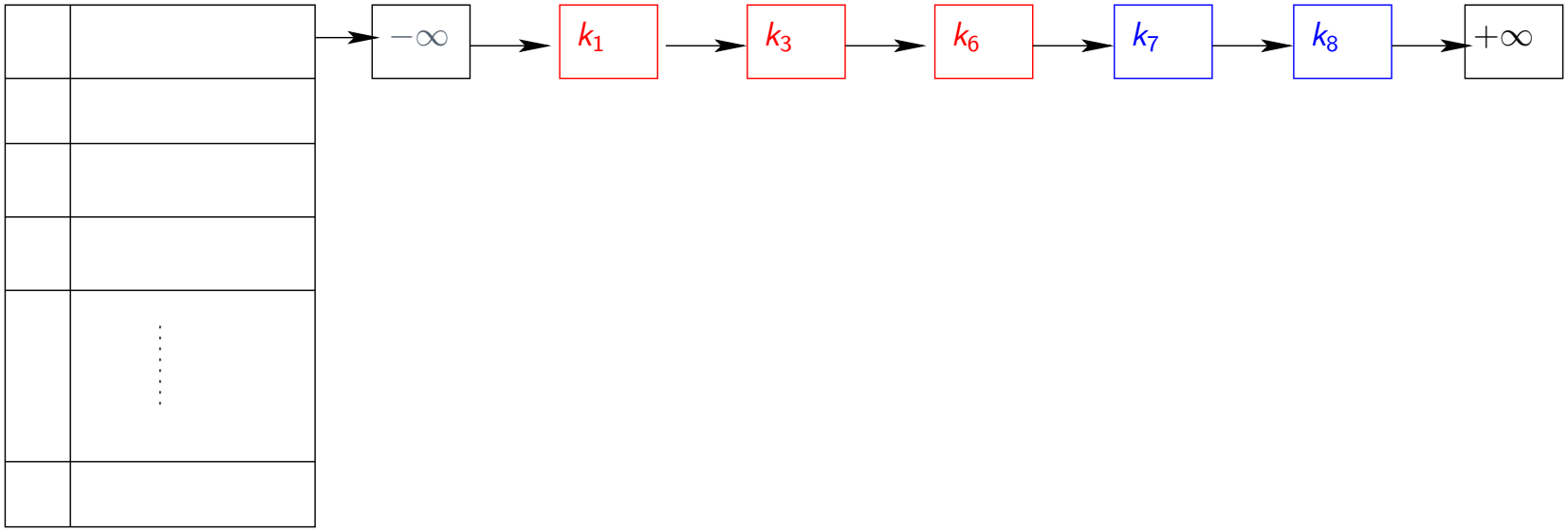

Figure: The underlying shared data-structure as hash table

HT-OSTM Exported methods:

STM_begin(), *STM_lookup()*, *STM_insert()*, *STM_delete()* and *STM_tryC()*

Challenge I

Maintaining Information of Deleted Nodes for Satisfying Opacity

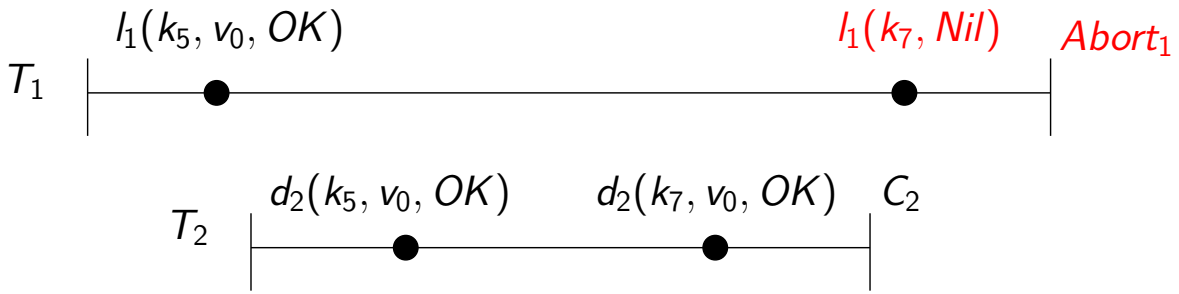

Figure: Time-Stamp of deleted node (k_7) is needed (Otherwise, not opaque).

Challenge I

Maintaining Information of Deleted Nodes for Satisfying Opacity

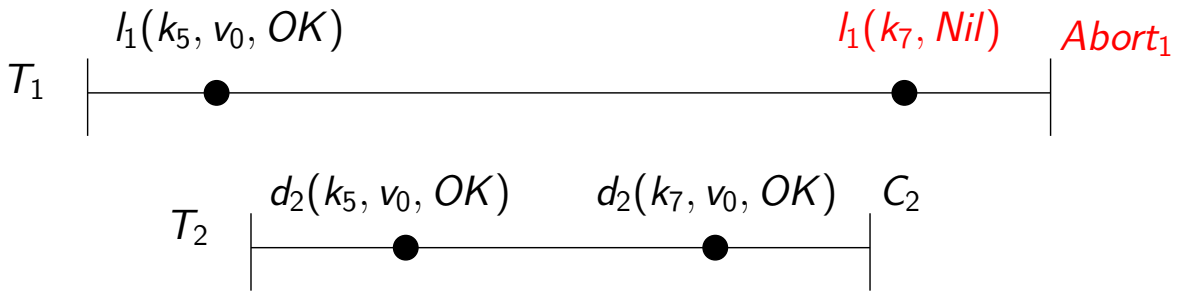

Figure: Time-Stamp of deleted node (k_7) is needed (Otherwise, not opaque).

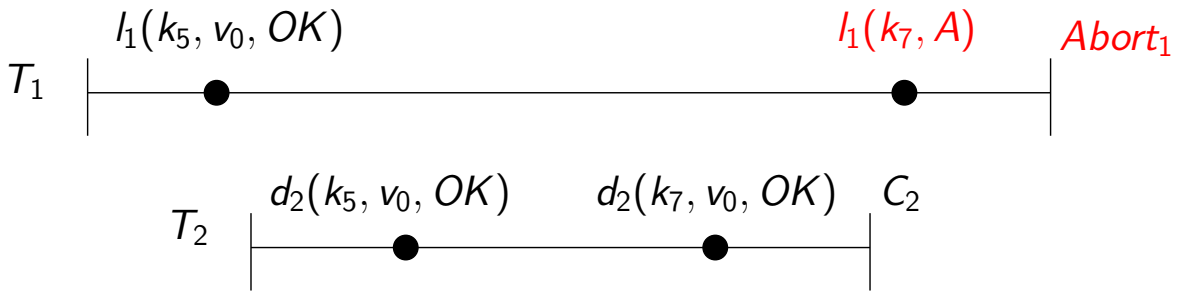

Figure: History is opaque.

Challenge I Contd..

Maintaining Information of Deleted Nodes

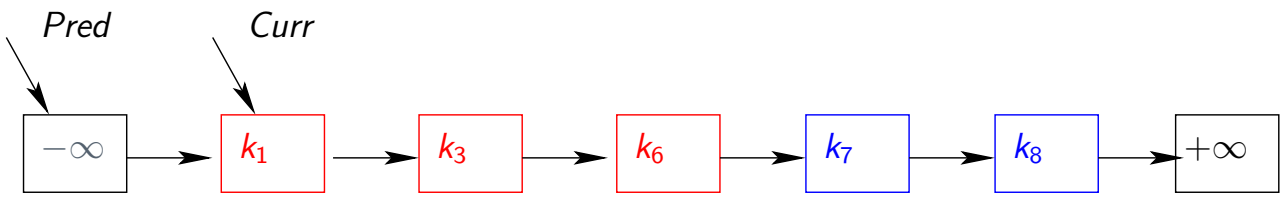

Figure: Issue: Searching key k_8 over lazylist

- Nodes are only logically deleted using **marked** field.
- Each node stores **time-stamp** along with key, value & marked field.
- Red color depicts dead node (deleted) and blue color depicts live node (not deleted).

Challenge I Contd..

Disadvantage of chaining with lazylist

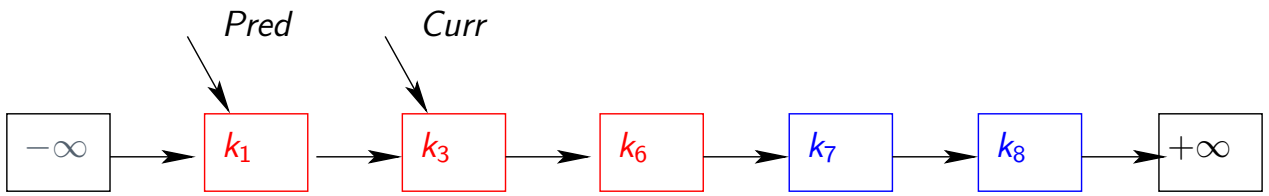

Figure: Issue: Searching key k_8 over lazylist

- Red color depicts dead node (deleted) and blue color depicts live node (not deleted).

Challenge I Contd..

Disadvantage of chaining with lazylist

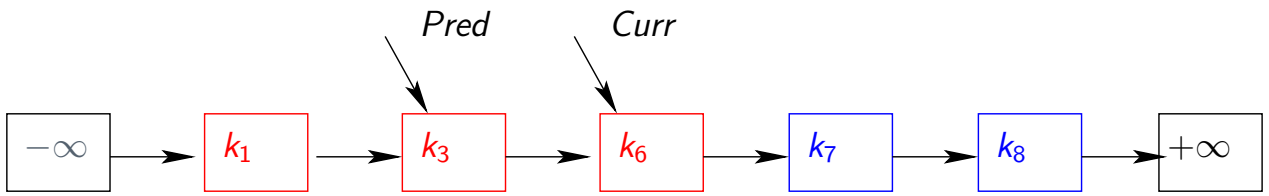

Figure: Issue: Searching key k_8 over lazylist

- Red color depicts dead node (deleted) and blue color depicts live node (not deleted).

Challenge I Contd..

Disadvantage of chaining with lazylist

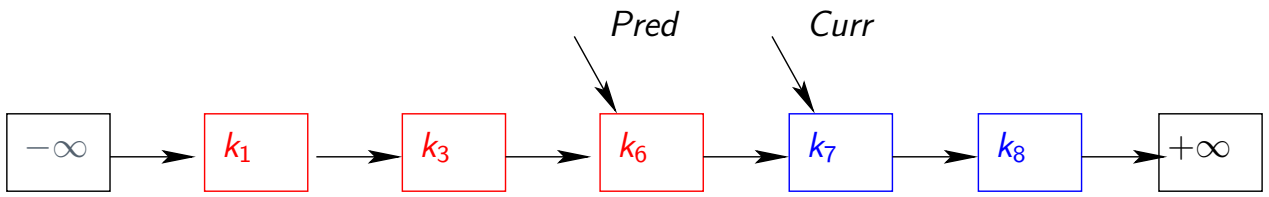

Figure: Issue: Searching key k_8 over lazylist

- Red color depicts dead node (deleted) and blue color depicts live node (not deleted).

Challenge I Contd..

Disadvantage of chaining with lazylist

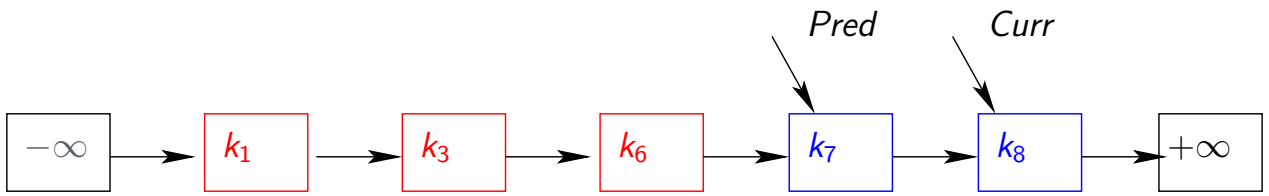

Figure: Issue: Searching key k_8 over lazylist

- Red color depicts dead node (deleted) and blue color depicts live node (not deleted).

Challenge I Contd..

Advantage of lazyrb-list over lazylist

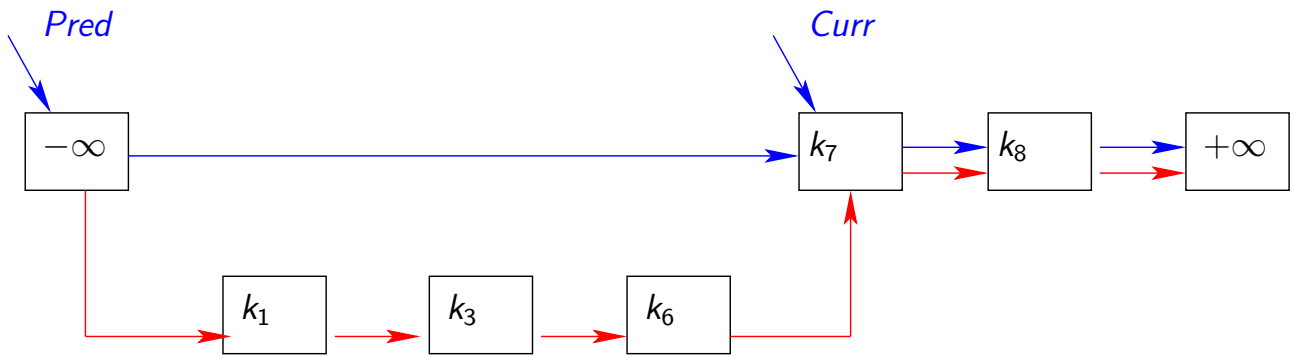

Figure: Searching key k_8 over lazyrb-list

- Each node is having two links: red link, blue link.
- Blue links are pointing to live nodes.
- Red links are pointing to live nodes as well as dead nodes.
- List invariants
  - Increasing order of keys.
  - Nodes accessible by blue links $\subseteq$ nodes accessible by red links.

Challenge I Contd..

Advantage of lazyrb-list over lazylist

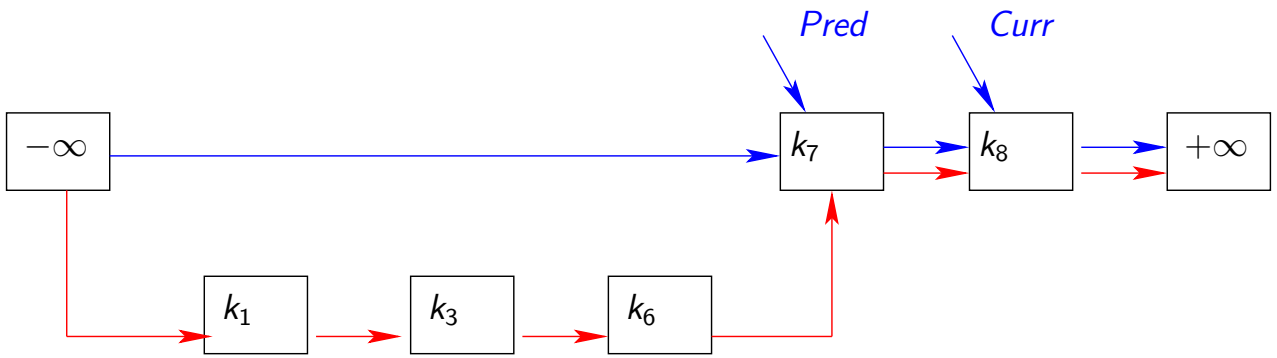

Figure: Searching key k_8 over lazyrb-list

- Each node is having two links: red link, blue link.
- Blue links are pointing to live nodes.
- Red links are pointing to live nodes as well as dead nodes.
- List invariants
  - Increasing order of keys.
  - Nodes accessible by blue links $\subseteq$ nodes accessible by red links.

Content

Execution Under HT-OSTM

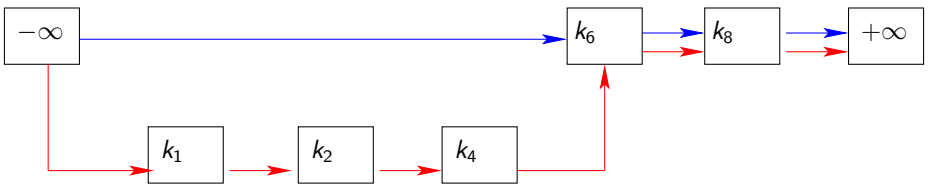

Figure: Underlying zoomed in lazyrb-list

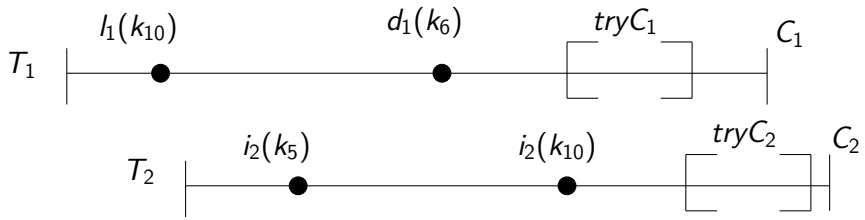

Figure: Example History

Execution Under HT-OSTM

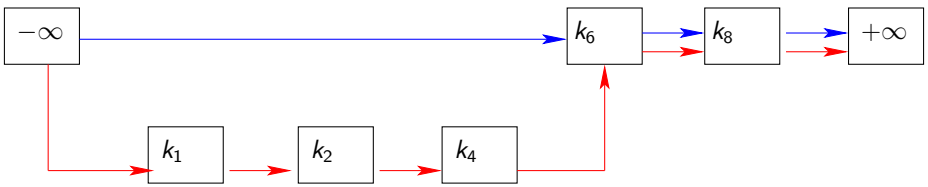

Figure: Underlying zoomed in lazyrb-list

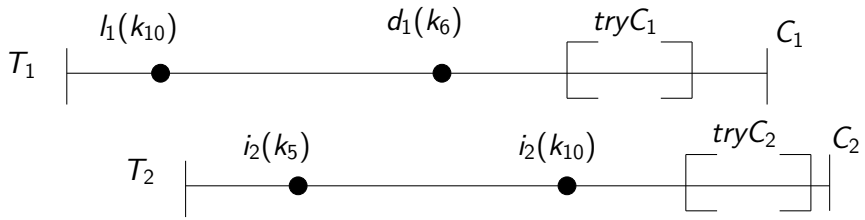

Figure: Example History

Figure: Current Execution

T_1

opn	key	P	C
lookup	k_{10}		

Figure: Transaction Log

Execution Under HT-OSTM

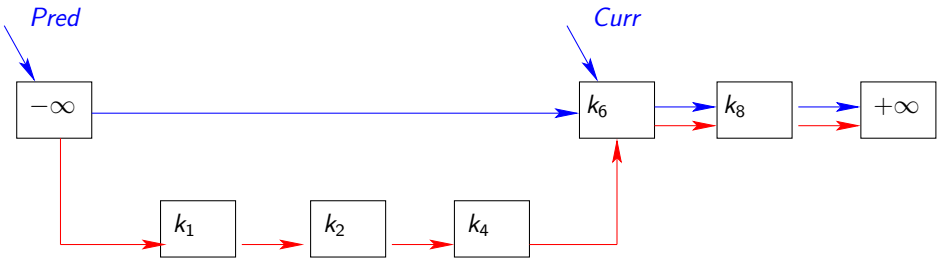

Figure: Underlying lazyrb-list: $l_1(k_{10})$ find region.

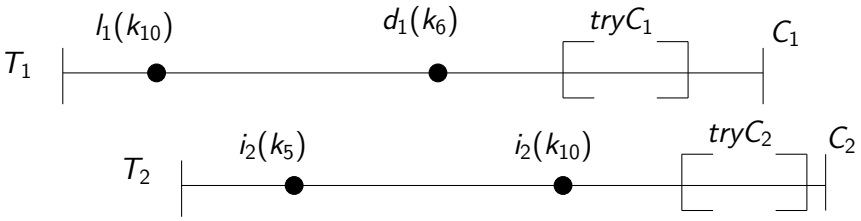

Figure: Example History

Figure: Current Execution

T_1

opn	key	P	C
lookup	k_{10}		

Figure: Transaction Log

Execution Under HT-OSTM

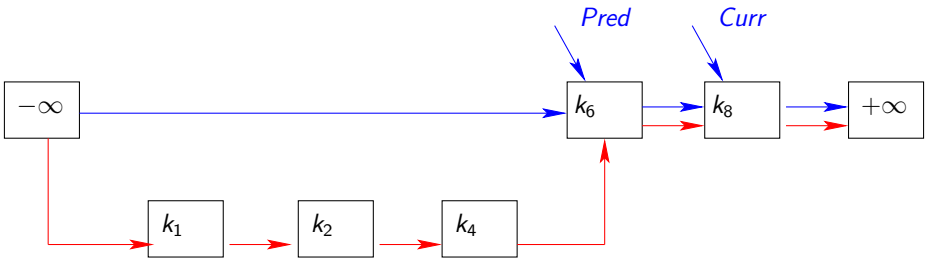

Figure: Underlying lazy-skip list: $l_1(k_{10})$ find region.

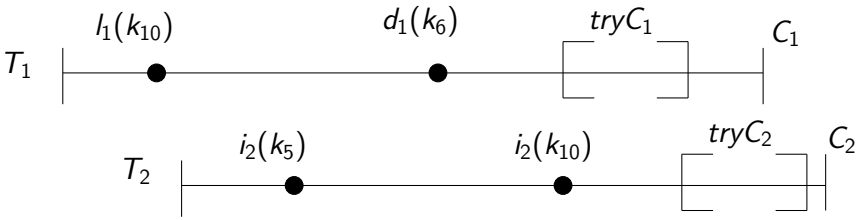

Figure: Example History

Figure: Current Execution

T_1

opn	key	P	C
lookup	k_{10}		

Figure: Transaction Log

Execution Under HT-OSTM

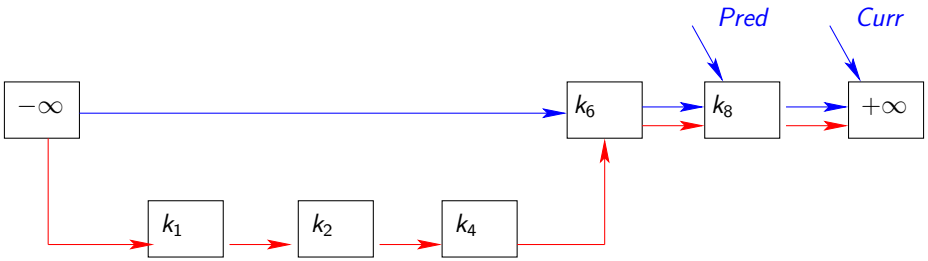

Figure: Underlying lazy-skip list: $l_1(k_{10})$ find region.

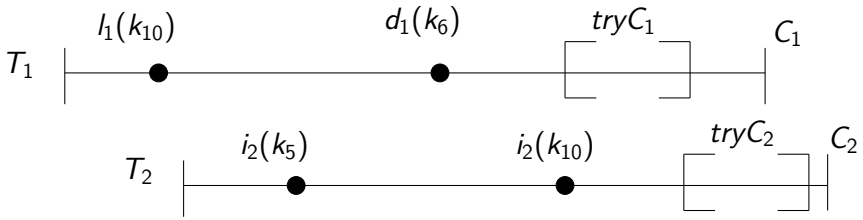

Figure: Example History

Figure: Current Execution

T_1

opn	key	P	C
lookup	k_{10}		

Figure: Transaction Log

Execution Under HT-OSTM

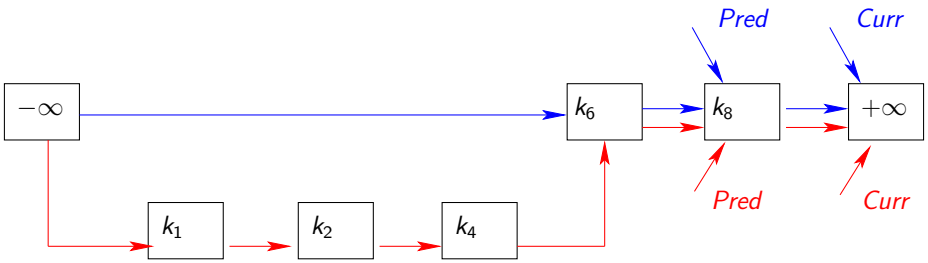

Figure: Underlying list: $l_1(k_{10})$: k_{10} not found.

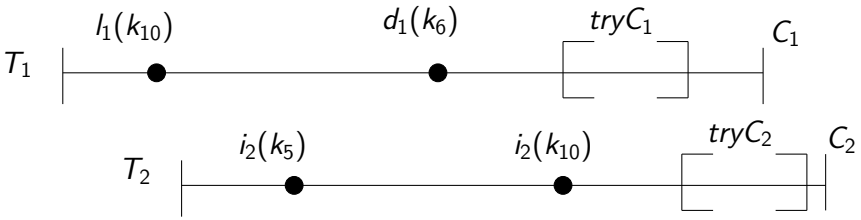

Figure: Example History

Figure: Current Execution

T_1

opn	key	P	C
lookup	k_{10}	k_8	$+\infty$
		k_8	$+\infty$

Figure: T_1: $l_1(k_{10})$: Log updated.

Execution Under HT-OSTM

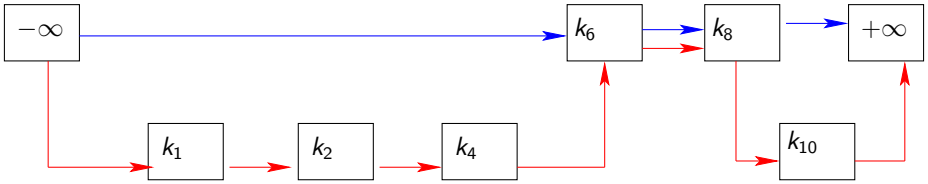

Figure: Underlying list: $l_1(k_{10})$: k_{10} added.

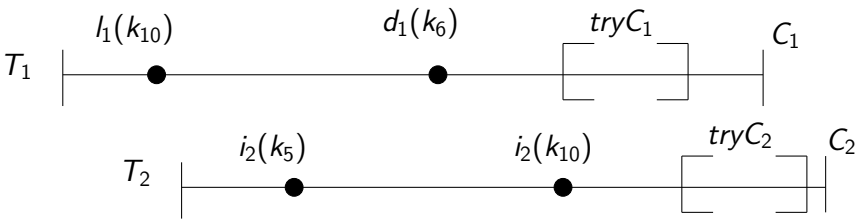

Figure: Example History

Figure: Current Execution

T_1

opn	key	P	C
lookup	k_{10}	k_8	$+\infty$
		k_8	$+\infty$

Figure: Transaction Log

Execution Under HT-OSTM

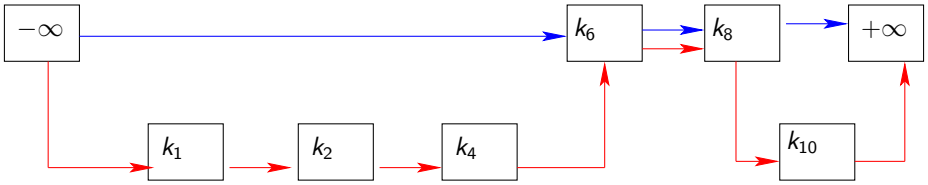

Figure: Underlying zoomed in lazy-skip list

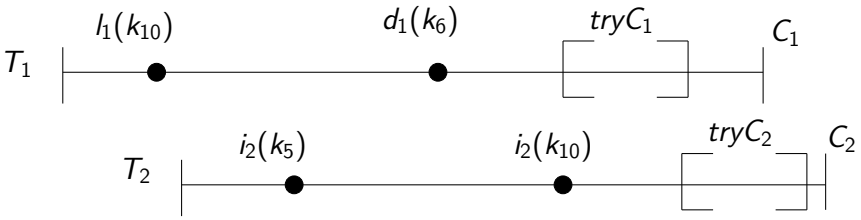

Figure: Example History

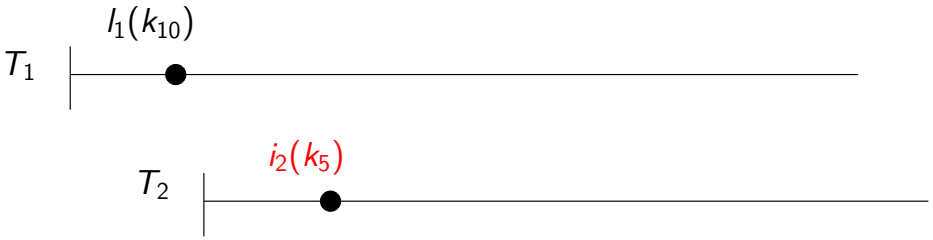

Figure: Current Execution

T_1

opn	key	P	C
lookup	k_{10}	k_8	$+\infty$
		k_8	$+\infty$

T_2

opn	key	P	C
insert	k_5		

Figure: T_2: $i_2(k_5)$: Log created.

Execution Under HT-OSTM

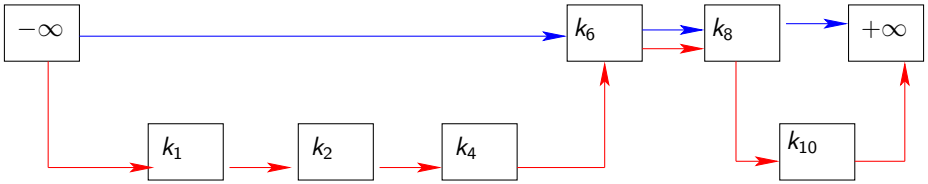

Figure: Underlying zoomed in lazy-skip list

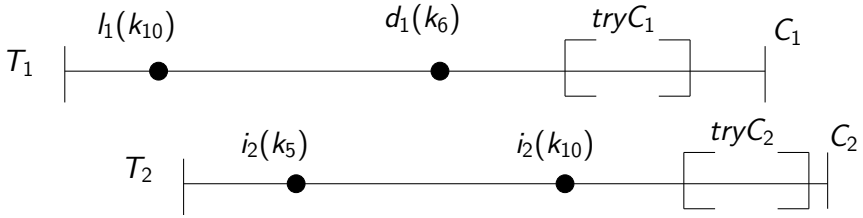

Figure: Example History

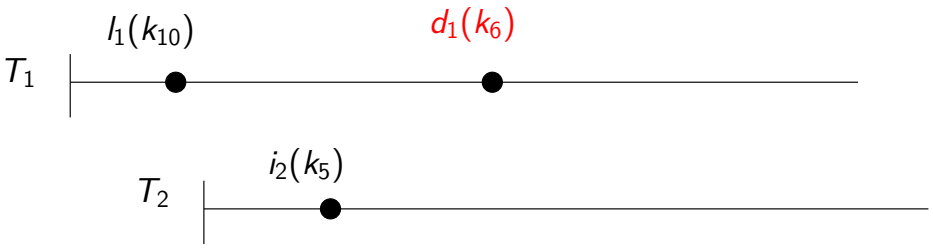

Figure: Current Execution

T_1

opn	key	P	C
lookup	k_{10}	k_8	$+\infty$
		k_8	$+\infty$

T_1

opn	key	P	C
delete	k_6		

T_2

opn	key	P	C
insert	k_5		

Figure: T_1: $d_1(k_6)$: Log created.

Execution Under HT-OSTM

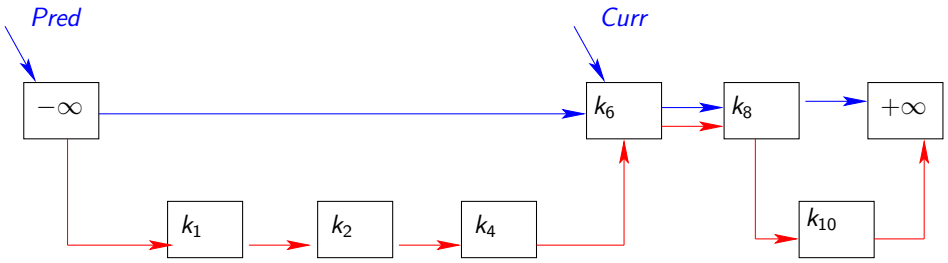

Figure: Underlying list: $d_1(k_6)$: k_6 find region.

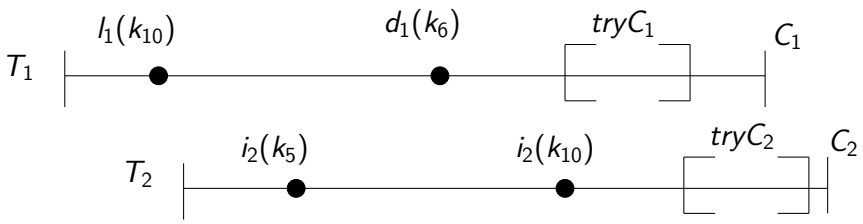

Figure: Example History

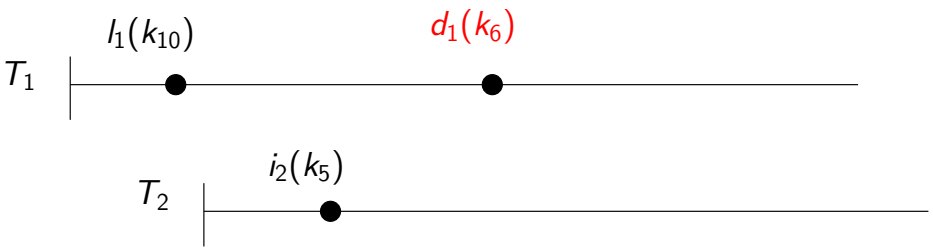

Figure: Current Execution

T_1

opn	key	P	C
lookup	k_{10}	k_8	$+\infty$
		k_8	$+\infty$

T_1

opn	key	P	C
delete	k_6		

T_2

opn	key	P	C
insert	k_5		

Figure: Transaction Log

Execution Under HT-OSTM

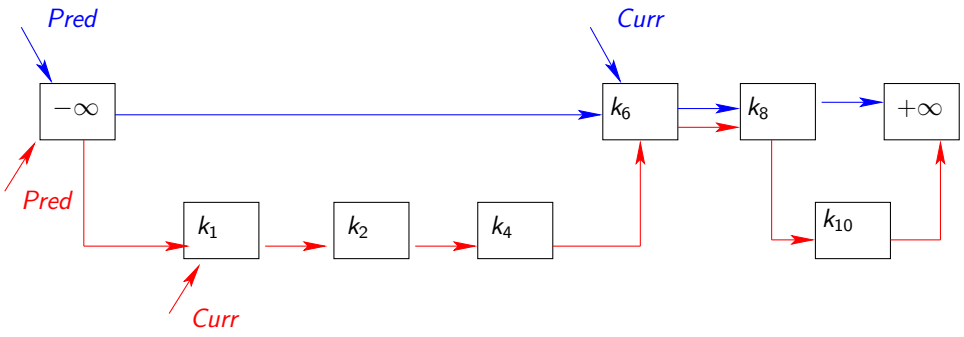

Figure: Underlying list: $d_1(k_6)$: k_6 find region.

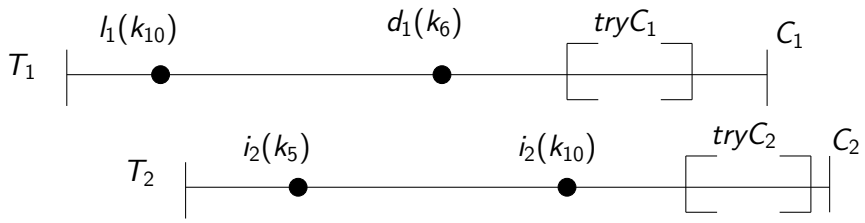

Figure: Example History

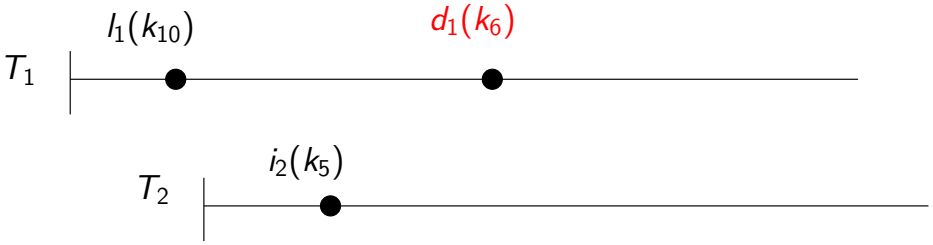

Figure: Current Execution

T_1

opn	key	P	C
lookup	k_{10}	k_8	$+\infty$
		k_8	$+\infty$

T_1

opn	key	P	C
delete	k_6		

T_2

opn	key	P	C
insert	k_5		

Figure: Transaction Log

Execution Under HT-OSTM

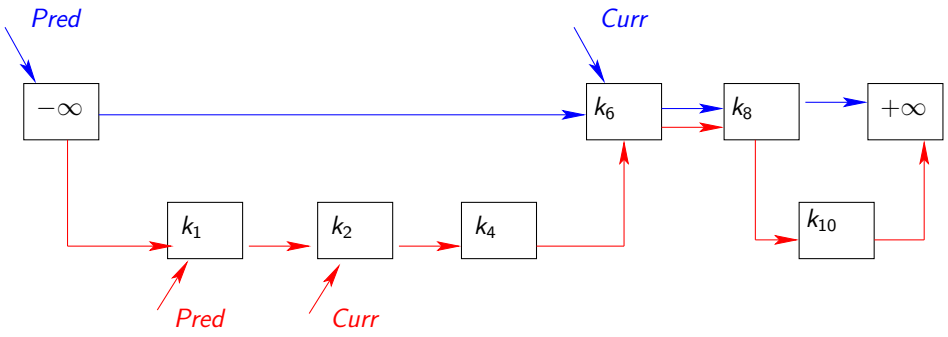

Figure: Underlying list: $d_1(k_6)$: k_6 find region.

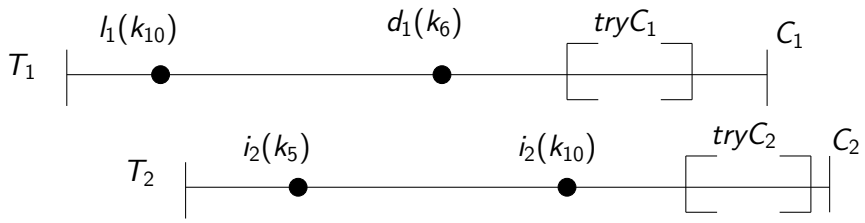

Figure: Example History

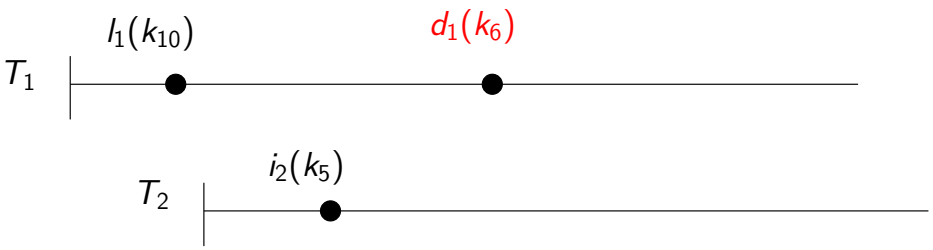

Figure: Current Execution

T_1

opn	key	P	C
lookup	k_{10}	k_8	$+\infty$
		k_8	$+\infty$

T_1

opn	key	P	C
delete	k_6		

T_2

opn	key	P	C
insert	k_5		

Figure: Transaction Log

Execution Under HT-OSTM

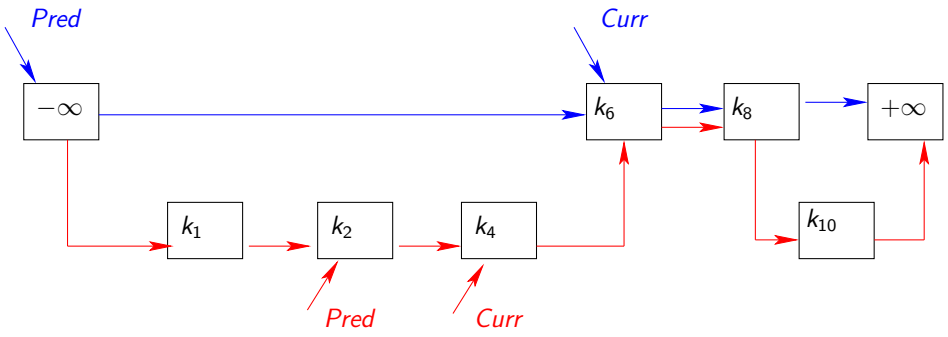

Figure: Underlying list: $d_1(k_6)$: k_6 find region.

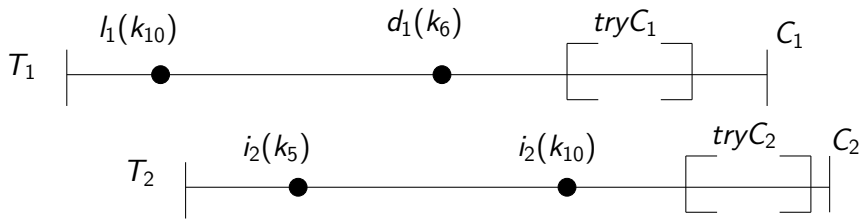

Figure: Example History

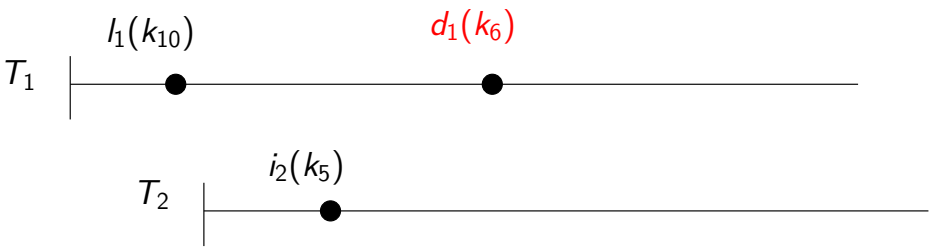

Figure: Current Execution

T_1

opn	key	P	C
lookup	k_{10}	k_8	$+\infty$
		k_8	$+\infty$

T_1

opn	key	P	C
delete	k_6		

T_2

opn	key	P	C
insert	k_5		

Figure: Transaction Log

Execution Under HT-OSTM

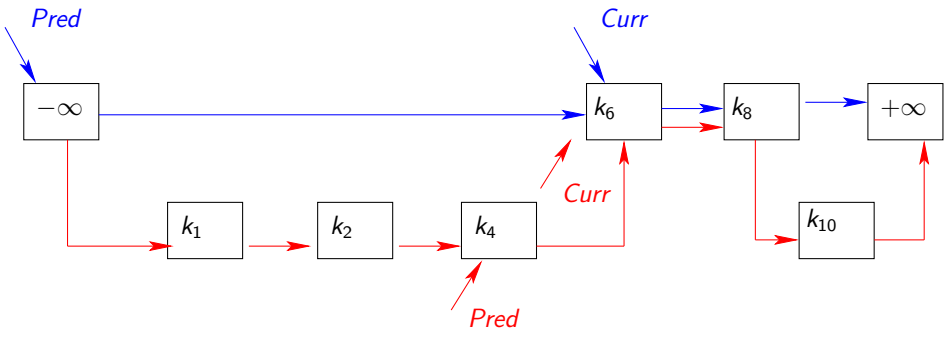

Figure: Underlying list: $d_1(k_6)$: k_6 found.

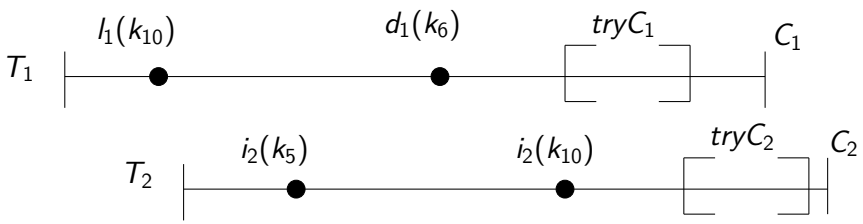

Figure: Example History

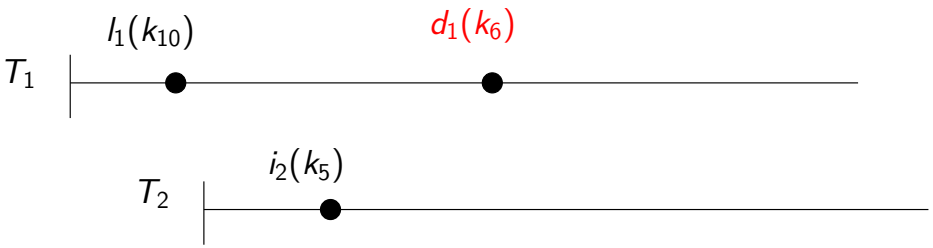

Figure: Current Execution

T_1

opn	key	P	C
lookup	k_{10}	k_8	$+\infty$
		k_8	$+\infty$

T_1

opn	key	P	C
delete	k_6	$-\infty$	k_6
		k_4	k_6

T_2

opn	key	P	C
insert	k_5		

Figure: T_1: $d_1(k_6)$ updated log.

Execution Under HT-OSTM

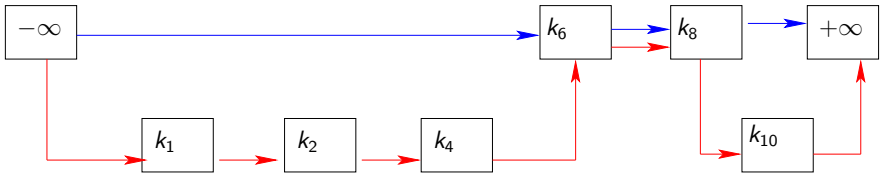

Figure: Underlying zoomed in lazy-skip list

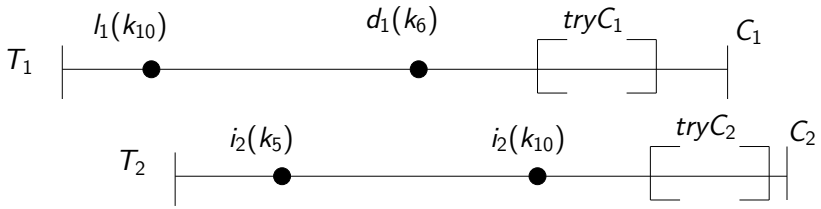

Figure: Example History

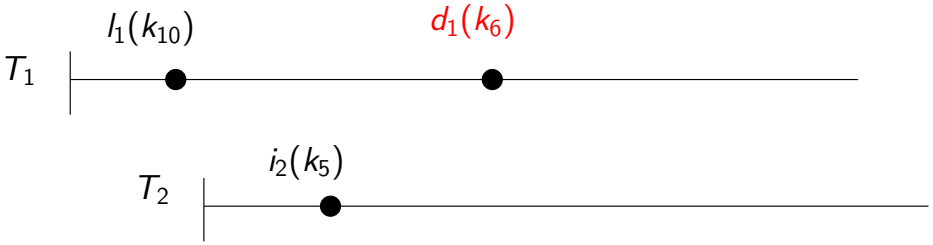

Figure: Current Execution

T_1

opn	key	P	C
lookup	k_{10}	k_8	$+\infty$
		k_8	$+\infty$

T_1

opn	key	P	C
delete	k_6	$-\infty$	k_6
		k_4	k_6

T_2

opn	key	P	C
insert	k_5		

Figure: Transaction Log

Execution Under HT-OSTM

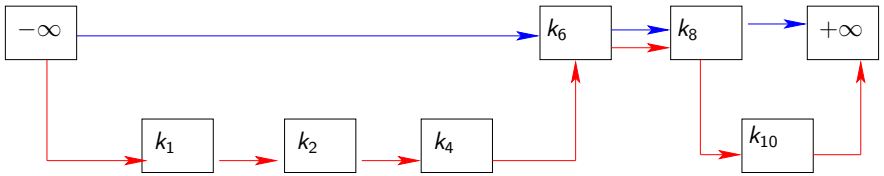

Figure: Underlying list: $i_2(k_{10})$: k_{10}.

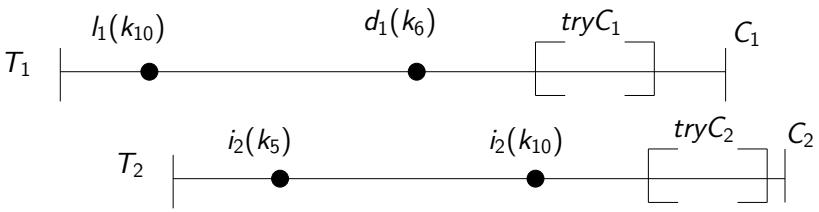

Figure: Example History

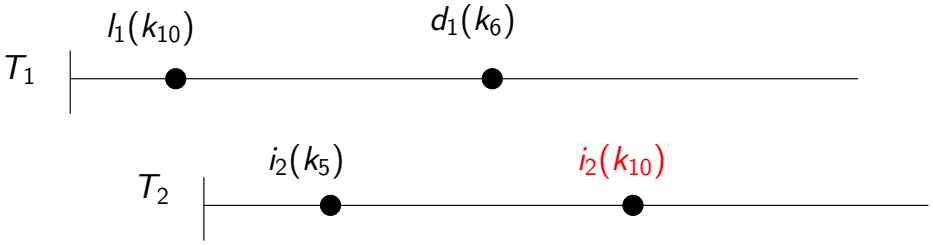

Figure: Current Execution

T_1

opn	key	P	C
lookup	k_{10}	k_8	$+\infty$
		k_8	$+\infty$

T_1

opn	key	P	C
delete	k_6	$-\infty$	k_6
		k_4	k_6

T_2

opn	key	P	C
insert	k_5		

T_2

opn	key	P	C
insert	k_{10}		

Figure: T_2: $i_2(k_{10})$ log created.

Execution Under HT-OSTM

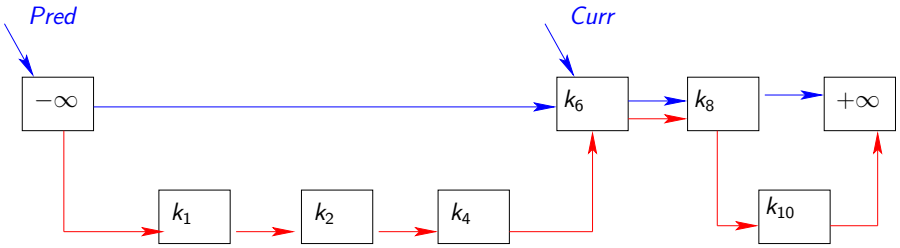

Figure: Underlying list: $tryC_1()$: $d_1(k_6)$.

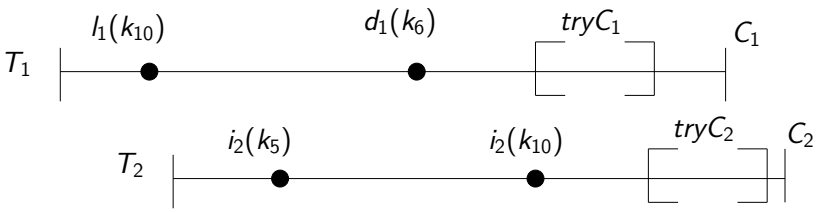

Figure: Example History

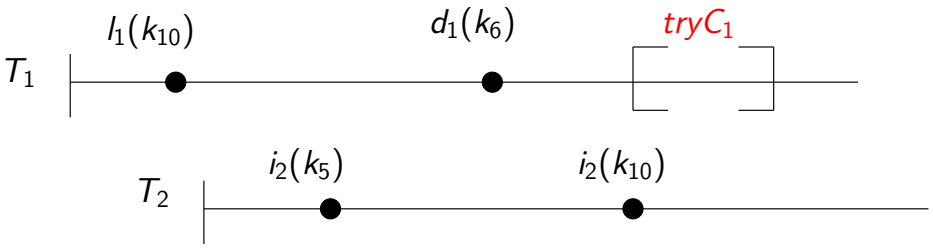

Figure: Current Execution

T_1

opn	key	P	C
lookup	k_{10}	k_8	$+\infty$
		k_8	$+\infty$

T_1

opn	key	P	C
delete	k_6	$-\infty$	k_6
		k_4	k_6

T_2

opn	key	P	C
insert	k_5		

T_2

opn	key	P	C
insert	k_{10}		

Figure: Transaction Log

Execution Under HT-OSTM

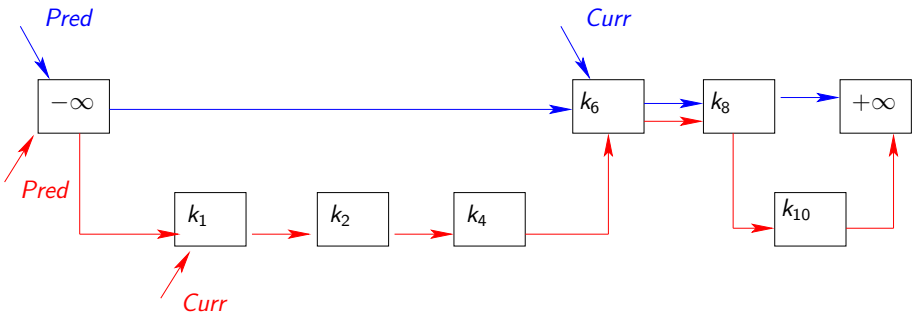

Figure: Underlying list: $tryC_1()$: $d_1(k_6)$ find region.

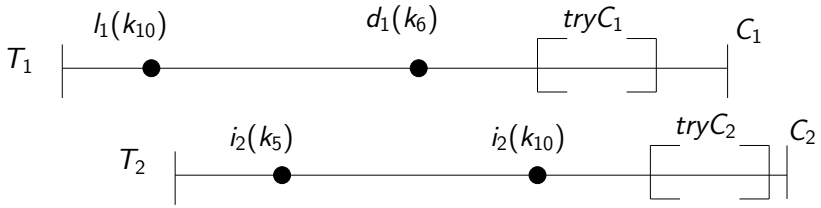

Figure: Example History

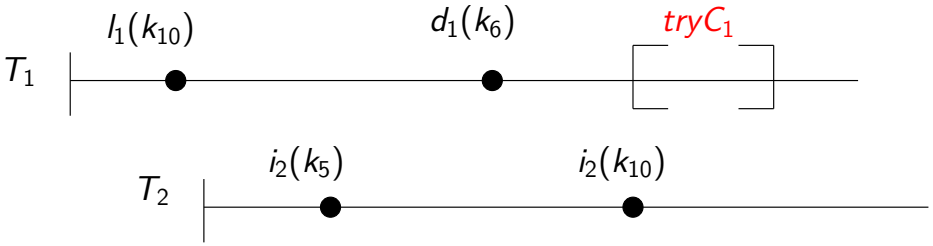

Figure: Current Execution

T_1

opn	key	P	C
lookup	k_{10}	k_8	$+\infty$
		k_8	$+\infty$

T_1

opn	key	P	C
delete	k_6	$-\infty$	k_6
		k_4	k_6

T_2

opn	key	P	C
insert	k_5		

T_2

opn	key	P	C
insert	k_{10}		

Figure: Transaction Log

Execution Under HT-OSTM

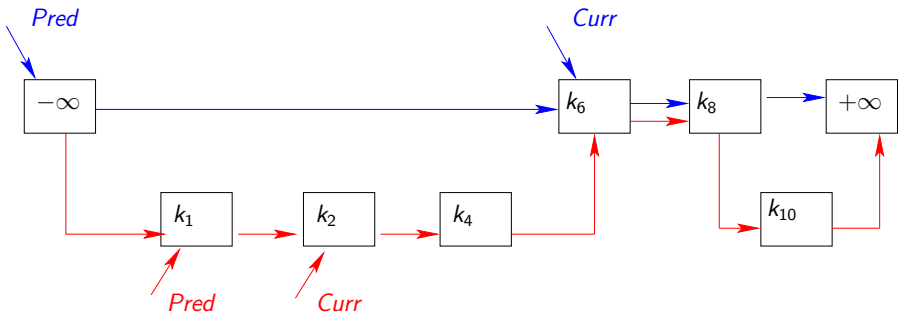

Figure: Underlying list: $tryC_1()$: $d_1(k_6)$ find region.

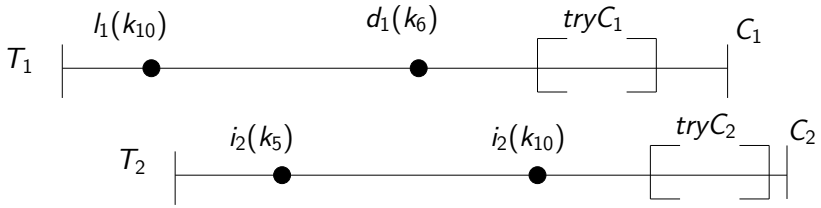

Figure: Example History

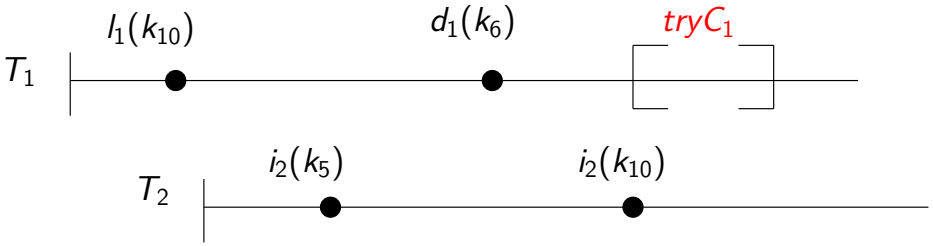

Figure: Current Execution

T_1

opn	key	P	C
lookup	k_{10}	k_8	$+\infty$
		k_8	$+\infty$

T_1

opn	key	P	C
delete	k_6	$-\infty$	k_6
		k_4	k_6

T_2

opn	key	P	C
insert	k_5		

T_2

opn	key	P	C
insert	k_{10}		

Figure: Transaction Log

Execution Under HT-OSTM

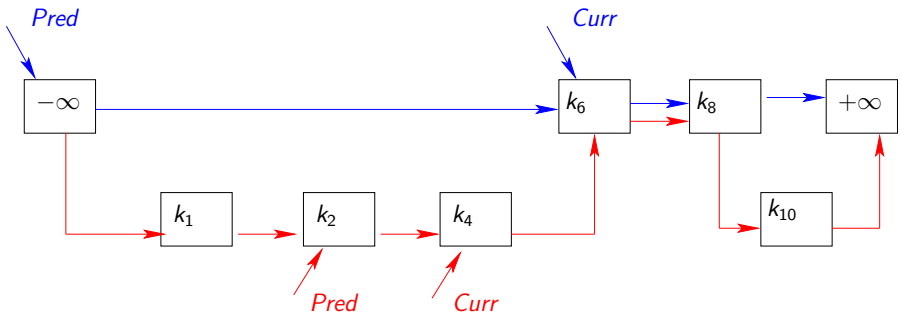

Figure: Underlying list: $tryC_1()$: $d_1(k_6)$ find region.

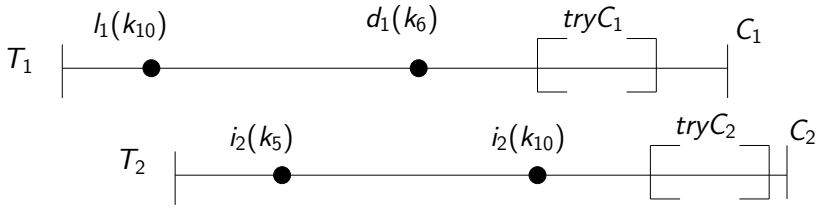

Figure: Example History

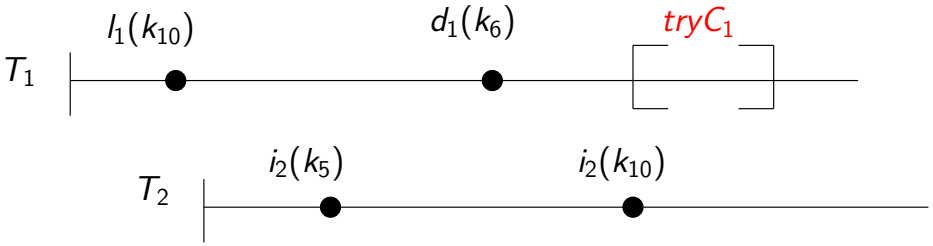

Figure: Current Execution

T_1

opn	key	P	C
lookup	k_{10}	k_8	$+\infty$
		k_8	$+\infty$

T_1

opn	key	P	C
delete	k_6	$-\infty$	k_6
		k_4	k_6

T_2

opn	key	P	C
insert	k_5		

T_2

opn	key	P	C
insert	k_{10}		

Figure: Transaction Log

Execution Under HT-OSTM

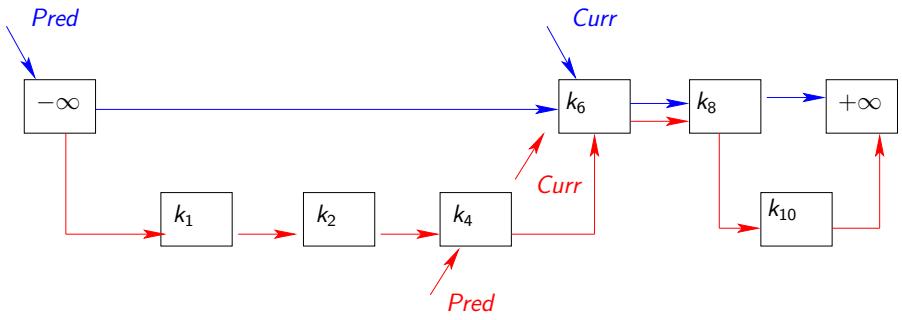

Figure: Underlying list: $tryC_1()$: $d_1(k_6)$ found.

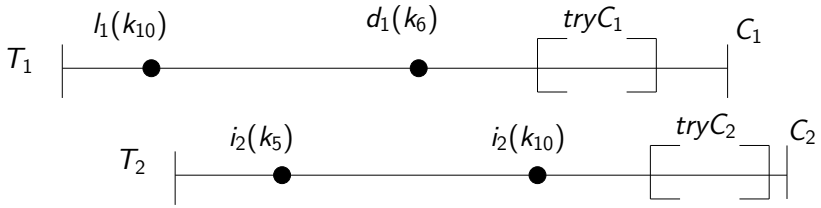

Figure: Example History

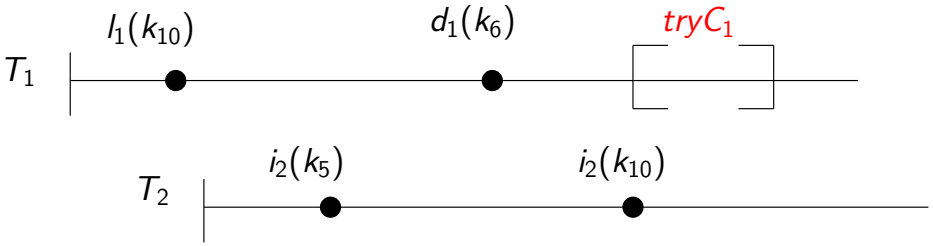

Figure: Current Execution

T_1

opn	key	P	C
lookup	k_{10}	k_8	$+\infty$
		k_8	$+\infty$

T_1

opn	key	P	C
delete	k_6	$-\infty$	k_6
		k_4	k_6

T_2

opn	key	P	C
insert	k_5		

T_2

opn	key	P	C
insert	k_{10}		

Figure: Transaction Log

Execution Under HT-OSTM

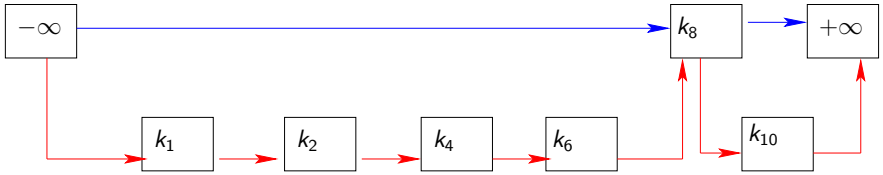

Figure: Underlying list: $tryC_1()$: $d_1(k_6)$ deleted.

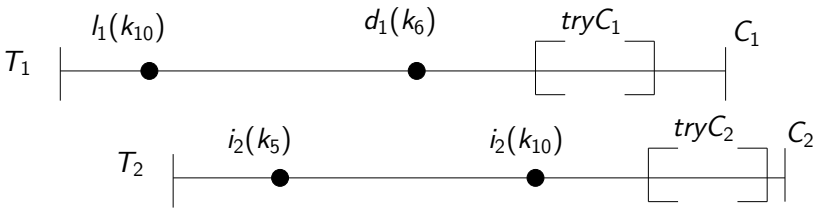

Figure: Example History

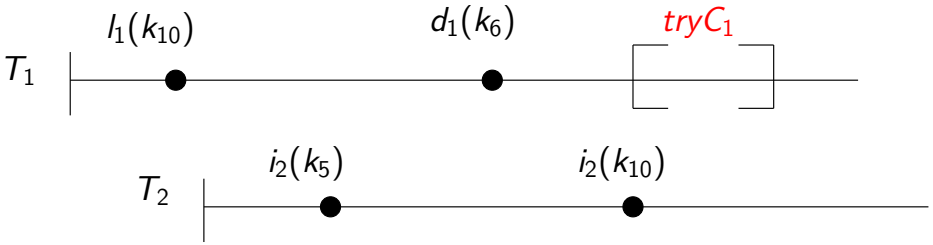

Figure: Current Execution

T_1

opn	key	P	C
lookup	k_{10}	k_8	$+\infty$
		k_8	$+\infty$

T_1

opn	key	P	C
delete	k_6	$-\infty$	k_6
		k_4	k_6

T_2

opn	key	P	C
insert	k_5		

T_2

opn	key	P	C
insert	k_{10}		

Figure: Transaction Log

Execution Under HT-OSTM

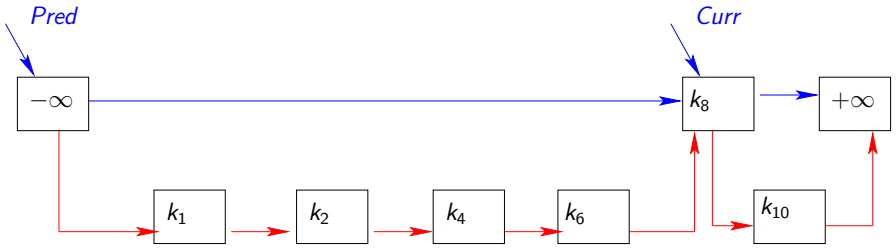

Figure: Underlying list: $tryC_2()$: $i_2(k_5)$ find region.

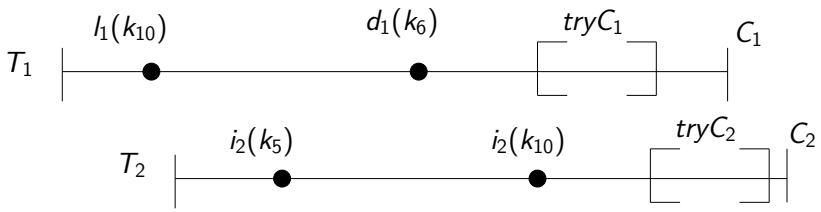

Figure: Example History

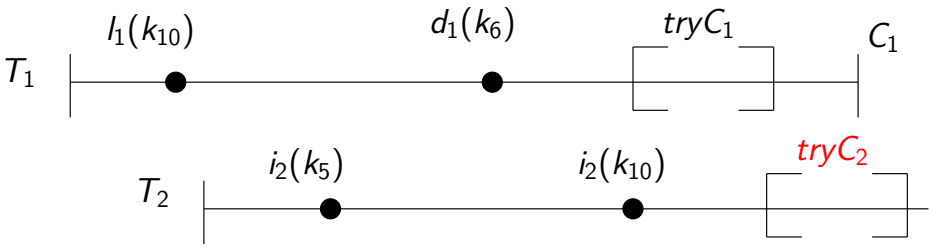

Figure: Current Execution

T_1

opn	key	P	C
lookup	k_{10}	k_8	$+\infty$
		k_8	$+\infty$

T_1

opn	key	P	C
delete	k_6	$-\infty$	k_6
		k_4	k_6

T_2

opn	key	P	C
insert	k_5		

T_2

opn	key	P	C
insert	k_{10}		

Figure: Transaction Log

Execution Under HT-OSTM

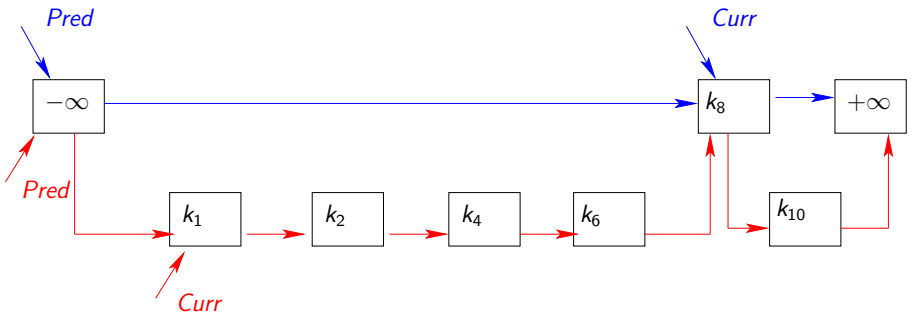

Figure: Underlying list: $tryC_2()$: $i_2(k_5)$ find region.

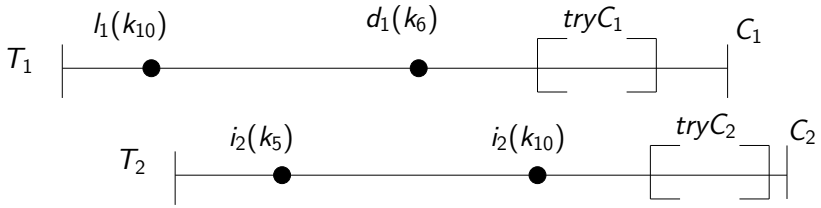

Figure: Example History

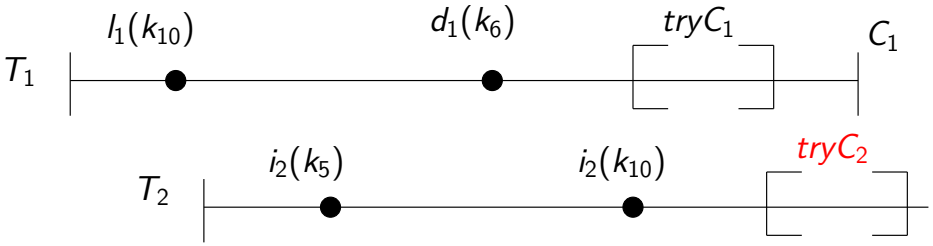

Figure: Current Execution

T_1

opn	key	P	C
lookup	k_{10}	k_8	$+\infty$
		k_8	$+\infty$

T_1

opn	key	P	C
delete	k_6	$-\infty$	k_6
		k_4	k_6

T_2

opn	key	P	C
insert	k_5		

T_2

opn	key	P	C
insert	k_{10}		

Figure: Transaction Log

Execution Under HT-OSTM

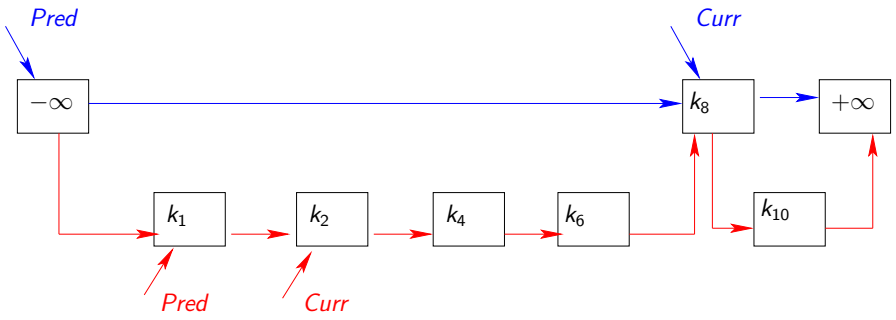

Figure: Underlying list: $tryC_2()$: $i_2(k_5)$ find region.

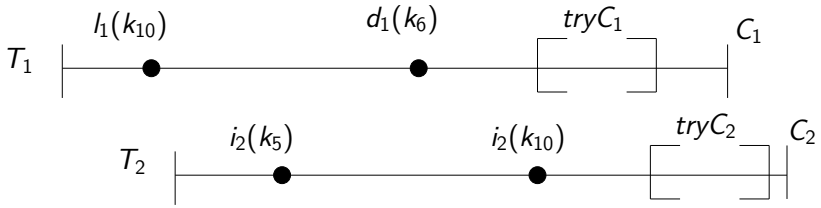

Figure: Example History

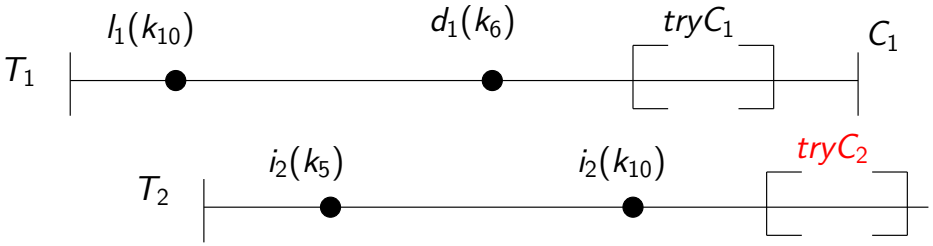

Figure: Current Execution

T_1

opn	key	P	C
lookup	k_{10}	k_8	$+\infty$
		k_8	$+\infty$

T_1

opn	key	P	C
delete	k_6	$-\infty$	k_6
		k_4	k_6

T_2

opn	key	P	C
insert	k_5		

T_2

opn	key	P	C
insert	k_{10}		

Figure: Transaction Log

Execution Under HT-OSTM

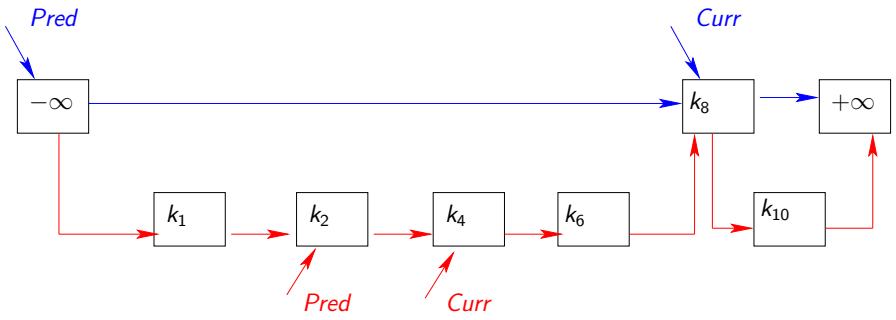

Figure: Underlying list: $tryC_2()$: $i_2(k_5)$ find region.

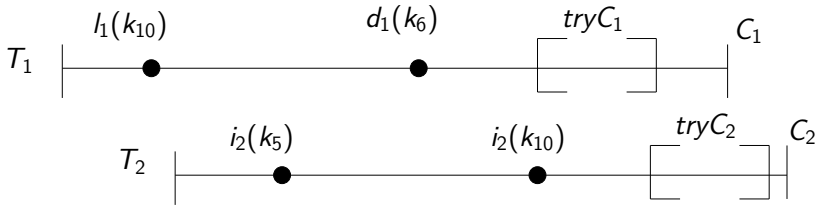

Figure: Example History

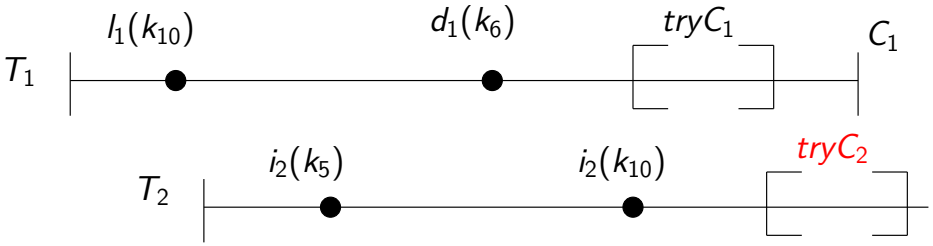

Figure: Current Execution

T_1

opn	key	P	C
lookup	k_{10}	k_8	$+\infty$
		k_8	$+\infty$

T_1

opn	key	P	C
delete	k_6	$-\infty$	k_6
		k_4	k_6

T_2

opn	key	P	C
insert	k_5		

T_2

opn	key	P	C
insert	k_{10}		

Figure: Transaction Log

Execution Under HT-OSTM

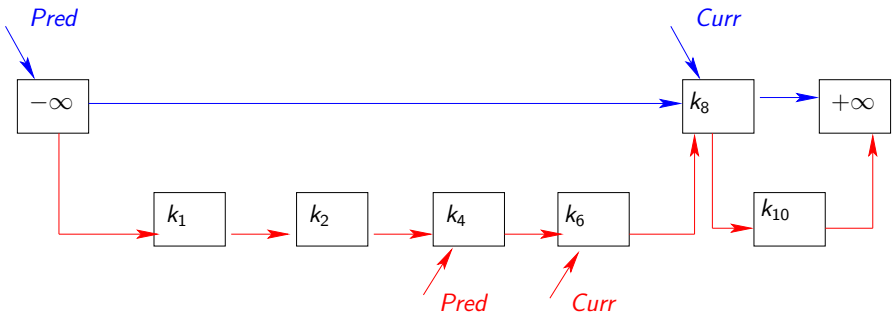

Figure: Underlying list: $tryC_2()$: $i_2(k_5)$ find region.

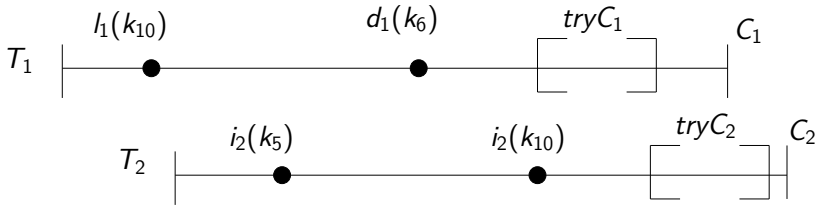

Figure: Example History

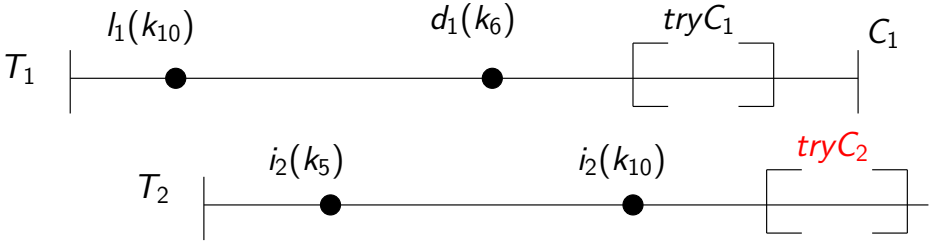

Figure: Current Execution

T_1

opn	key	P	C
lookup	k_{10}	k_8	$+\infty$
		k_8	$+\infty$

T_1

opn	key	P	C
delete	k_6	$-\infty$	k_6
		k_4	k_6

T_2

opn	key	P	C
insert	k_5	$-\infty$	k_8
		k_4	k_6

T_2

opn	key	P	C
insert	k_{10}		

Figure: $i_2(k_5)$ log updated.

Execution Under HT-OSTM

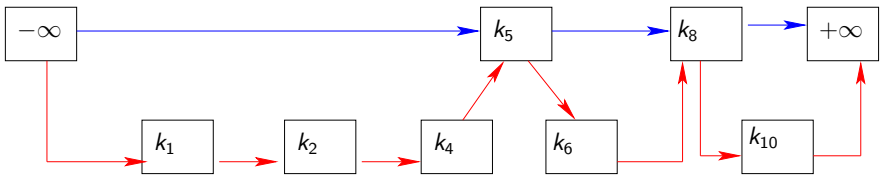

Figure: Underlying list: $tryC_2()$: k_5 added.

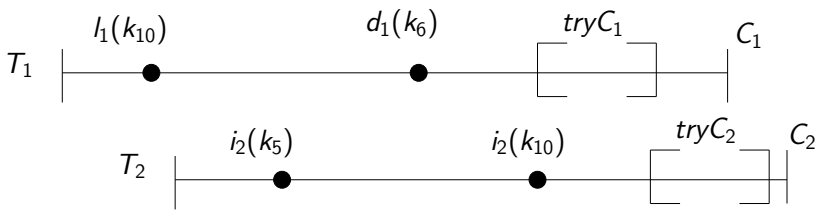

Figure: Example History

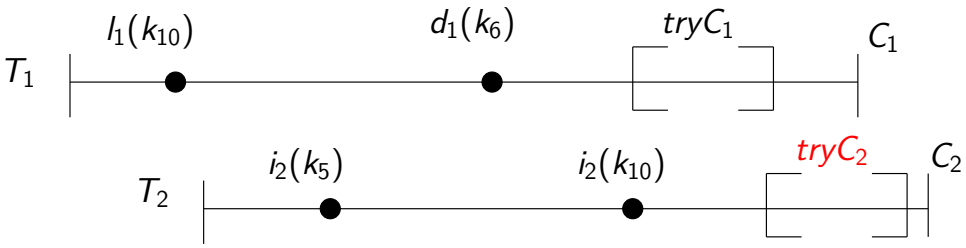

Figure: Current Execution

T_1

opn	key	P	C
lookup	k_{10}	k_8	$+\infty$
		k_8	$+\infty$

T_1

opn	key	P	C
delete	k_6	$-\infty$	k_6
		k_4	k_6

T_2

opn	key	P	C
insert	k_5	$-\infty$	k_8
		k_4	k_6

T_2

opn	key	P	C
insert	k_{10}		

Figure: Transaction Log

Execution Under HT-OSTM

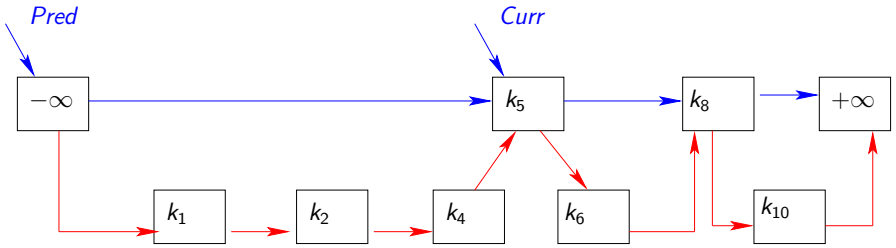

Figure: Underlying list: $tryC_2()$: $i_2(k_{10})$ find region.

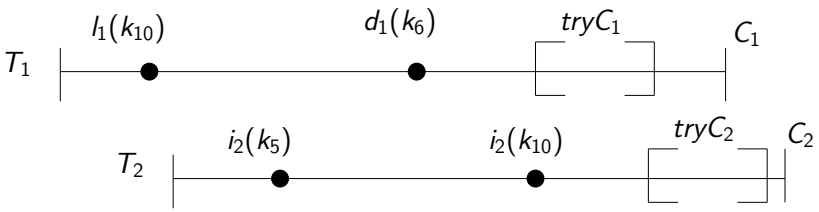

Figure: Example History

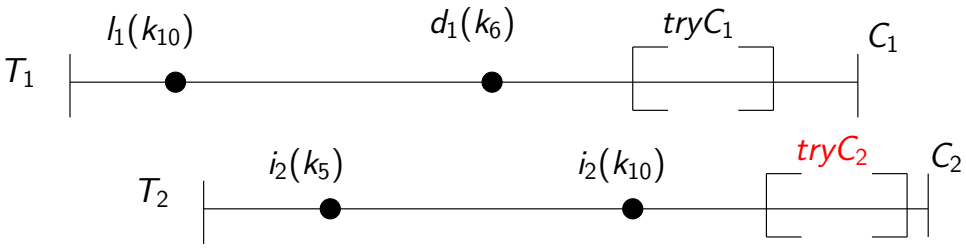

Figure: Current Execution

T_1

opn	key	P	C
lookup	k_{10}	k_8	$+\infty$
		k_8	$+\infty$

T_1

opn	key	P	C
delete	k_6	$-\infty$	k_6
		k_4	k_6

T_2

opn	key	P	C
insert	k_5	$-\infty$	k_8
		k_4	k_6

T_2

opn	key	P	C
insert	k_{10}		

Figure: Transaction Log

Execution Under HT-OSTM

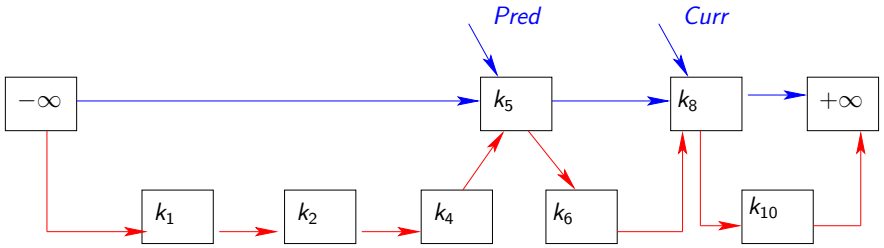

Figure: Underlying list: $tryC_2()$: $i_2(k_{10})$ find region.

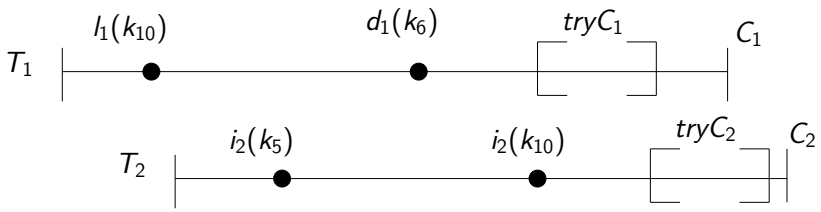

Figure: Example History

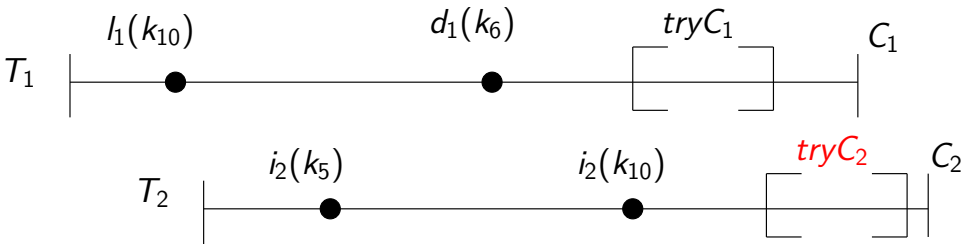

Figure: Current Execution

T_1

opn	key	P	C
lookup	k_{10}	k_8	$+\infty$
		k_8	$+\infty$

T_1

opn	key	P	C
delete	k_6	$-\infty$	k_6
		k_4	k_6

T_2

opn	key	P	C
insert	k_5	$-\infty$	k_8
		k_4	k_6

T_2

opn	key	P	C
insert	k_{10}		

Figure: Transaction Log

Execution Under HT-OSTM

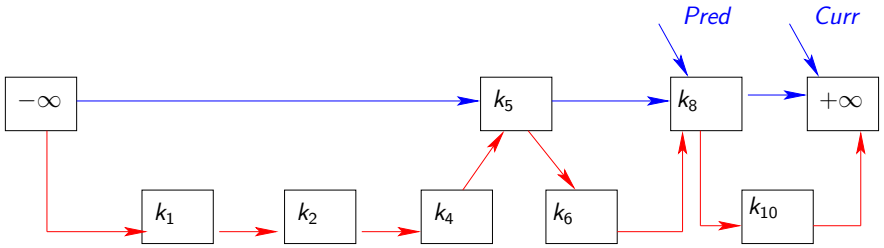

Figure: Underlying list: $tryC_2()$: $i_2(k_{10})$ find region.

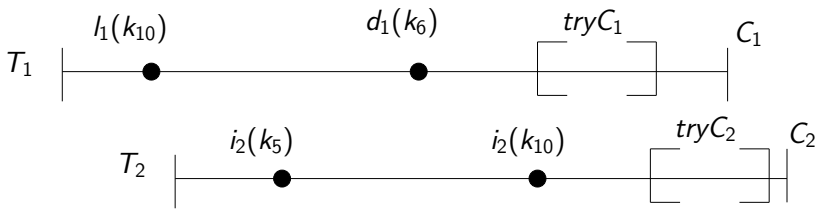

Figure: Example History

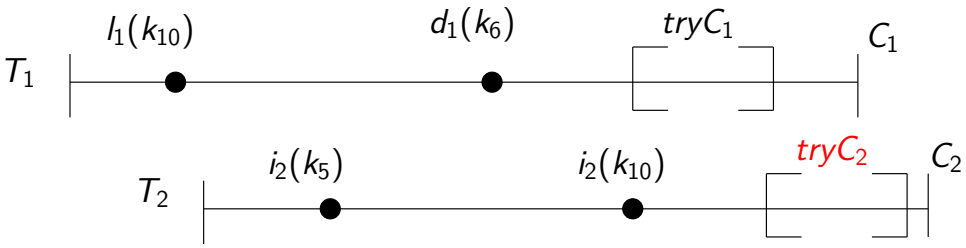

Figure: Current Execution

T_1

opn	key	P	C
lookup	k_{10}	k_8	$+\infty$
		k_8	$+\infty$

T_1

opn	key	P	C
delete	k_6	$-\infty$	k_6
		k_4	k_6

T_2

opn	key	P	C
insert	k_5	$-\infty$	k_8
		k_4	k_6

T_2

opn	key	P	C
insert	k_{10}		

Figure: Transaction Log

Execution Under HT-OSTM

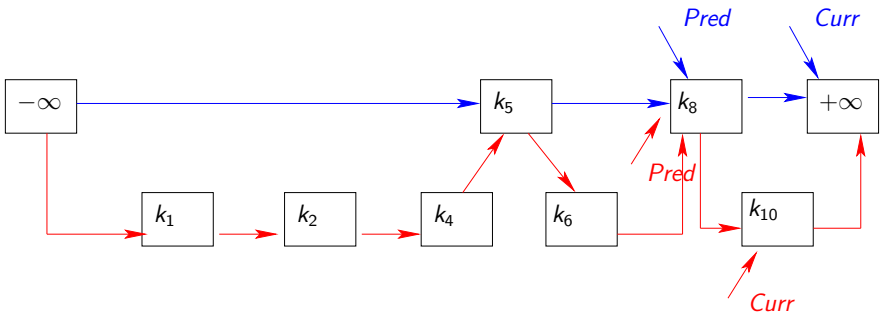

Figure: Underlying list: $tryC_2()$: $i_2(k_{10})$ find region.

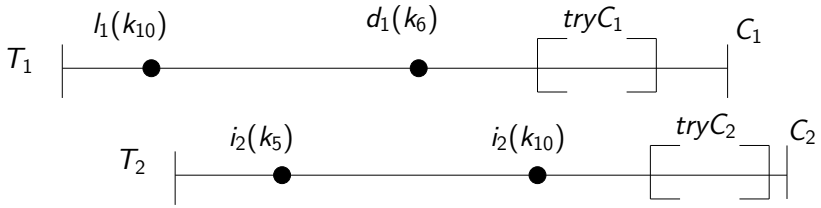

Figure: Example History

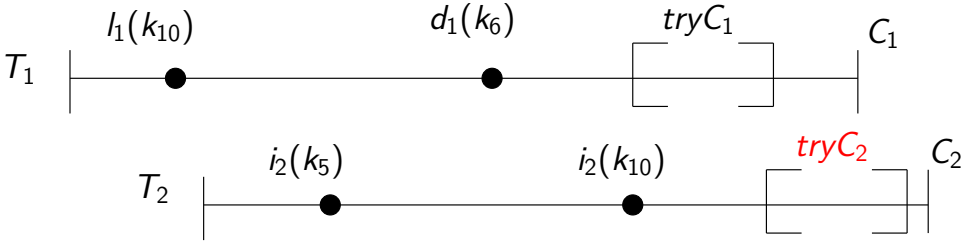

Figure: Current Execution

T_1

opn	key	P	C
lookup	k_{10}	k_8	$+\infty$
		k_8	$+\infty$

T_1

opn	key	P	C
delete	k_6	$-\infty$	k_6
		k_4	k_6

T_2

opn	key	P	C
insert	k_5	$-\infty$	k_8
		k_4	k_6

T_2

opn	key	P	C
insert	k_{10}	k_8	$+\infty$
		k_8	k_{10}

Figure: $i_2(k_{10})$ updated log.

Execution Under HT-OSTM

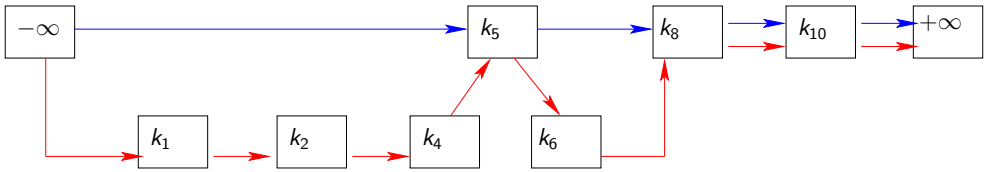

Figure: Underlying list: $tryC_2()$: k_{10} added

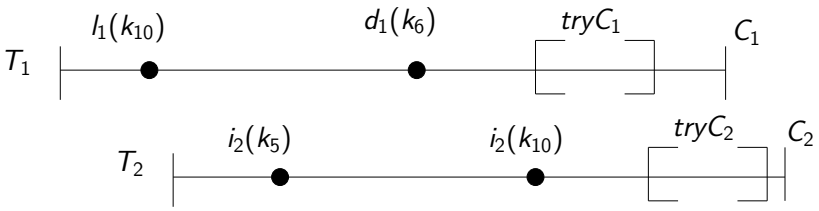

Figure: Example History

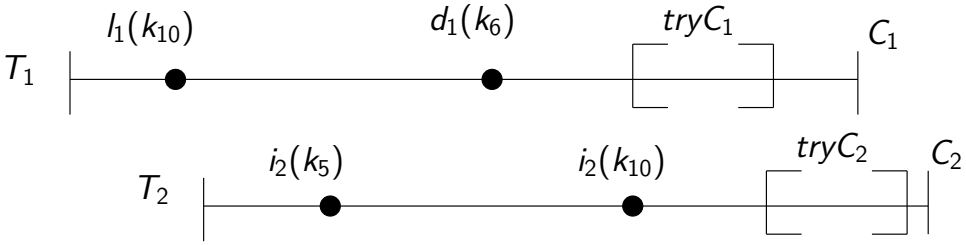

Figure: Current Execution

T_1

opn	key	P	C
lookup	k_{10}	k_8	$+\infty$
		k_8	$+\infty$

T_1

opn	key	P	C
delete	k_6	$-\infty$	k_6
		k_4	k_6

T_2

opn	key	P	C
insert	k_5	$-\infty$	k_8
		k_4	k_6

T_2

opn	key	P	C
insert	k_{10}	k_8	$+\infty$
		k_8	k_{10}

Figure: Transaction Log

Content

Proof Of Correctness

Operation level and Transaction level

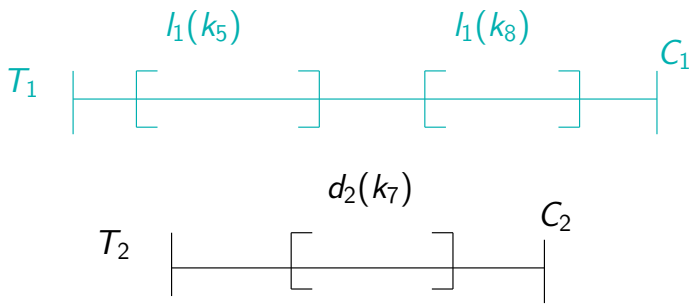

Figure: Overlapping Operations

Proof Of Correctness

Operation level and Transaction level

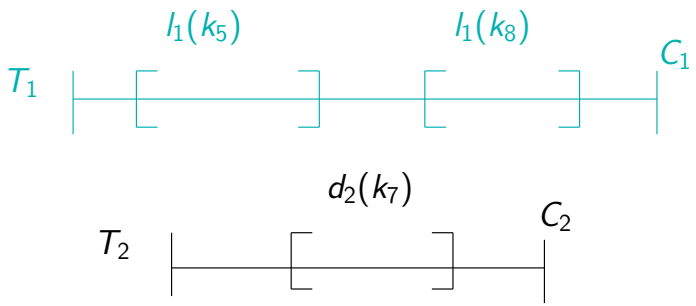

Figure: Overlapping Operations

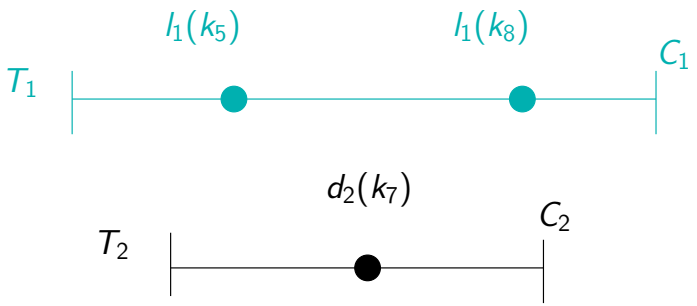

Figure: Operations are Linearized

Proof Of Correctness

Operation level and Transaction level

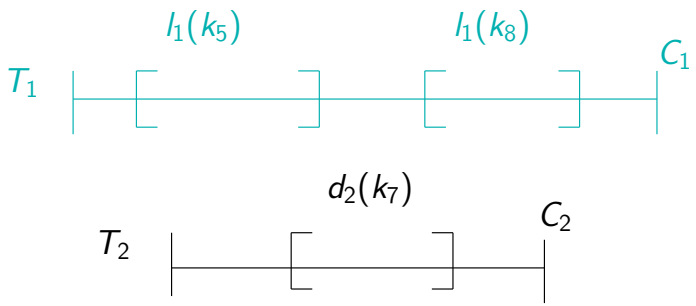

Figure: Overlapping Operations

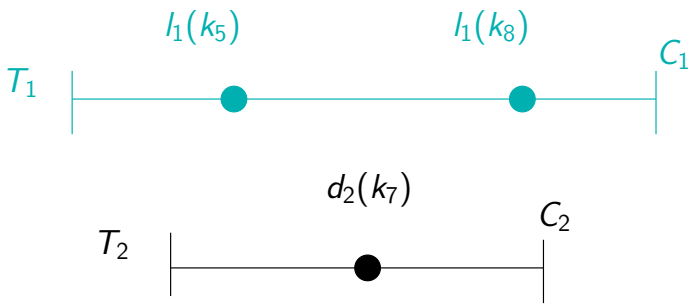

Figure: Operations are Linearized

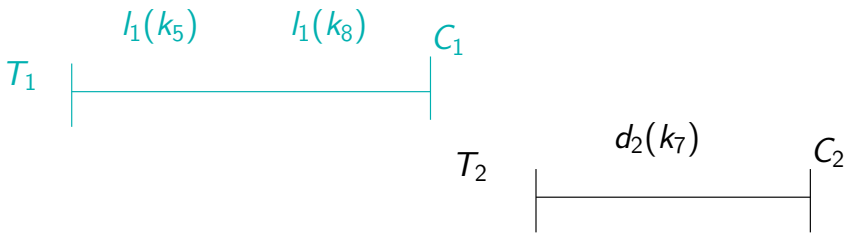

Figure: Serial History (T_1, T_2)

Proof Of Correctness

Operation level and Transaction level

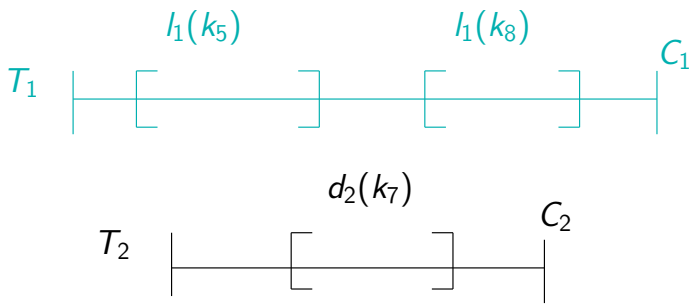

Figure: Overlapping Operations

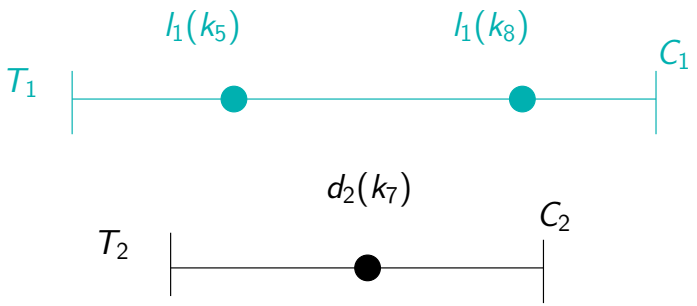

Figure: Operations are Linearized

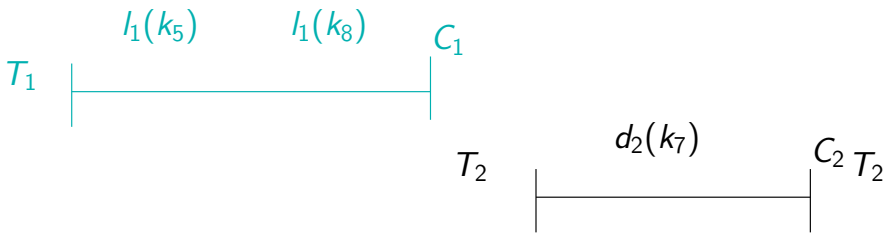

Figure: Serial History (T_1, T_2)

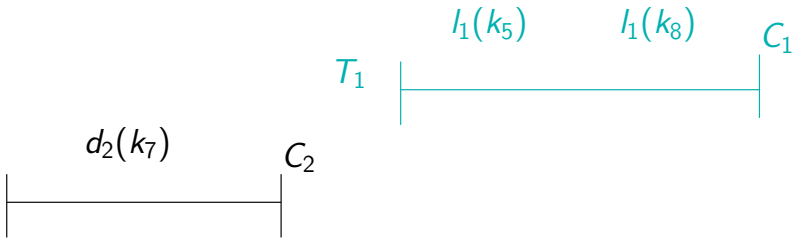

Figure: Serial History (T_2, T_1)

Proof Of Correctness

Theorem (1)

Consider a history H generated by HT-OSTM, there exists a sequential & legal history H' equivalent to H such that the conflict-graph of H' is acyclic.

Proof Of Correctness

Theorem (1)

Consider a history H generated by HT-OSTM, there exists a sequential & legal history H' equivalent to H such that the conflict-graph of H' is acyclic.

Theorem (2)

A legal HT-OSTM history H is opaque iff $CG(H)$ is acyclic.

Please refer the arxiv link—> `https://arxiv.org/abs/1709.00681` for more detials.

Content

Experimental Setup

Setup

- Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz, 56 NUMA CPUs.
- HT-OSTM vs Basic Time stamp ordering Protocol(BTO) [Weikum et al.] / Elastic STM(ESTM) [Gramoli et al.].
- list-OSTM vs Boosted list(BST) [Herlihy et al.] / NOrec STM list(NTM) [Michael et al.] / lock-free transactional list(LFT) [Zhang et al.]
- Time/thread and Number of aborts/thread

Experimental Setup

Setup

- Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz, 56 NUMA CPUs.
- HT-OSTM vs Basic Time stamp ordering Protocol(BTO) [Weikum et al.] / Elastic STM(ESTM) [Gramoli et al.].
- list-OSTM vs Boosted list(BST) [Herlihy et al.] / NOrec STM list(NTM) [Michael et al.] / lock-free transactional list(LFT) [Zhang et al.]
- Time/thread and Number of aborts/thread

Parameters

- Lookup Intensive Workload: lookup:70%, insert:10% & delete:20%
- Update Intensive Workload: lookup:50%, insert:25% & delete:25%
- Key range: 1000 and Operations/transaction: 10

Results
HT-OSTM against ESTM, RWSTM

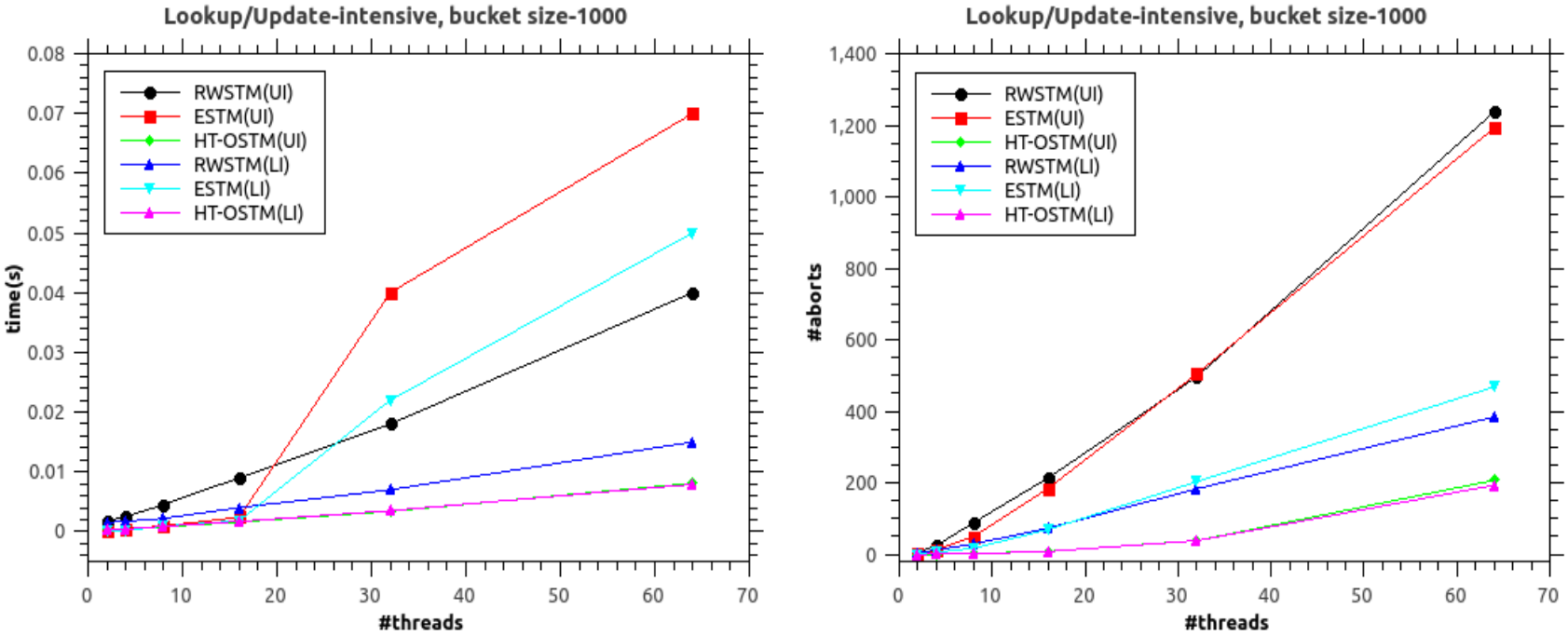

Figure: time vs #threads OR #aborts vs #threads

Results

list-OSTM against LTM, NTM and BST

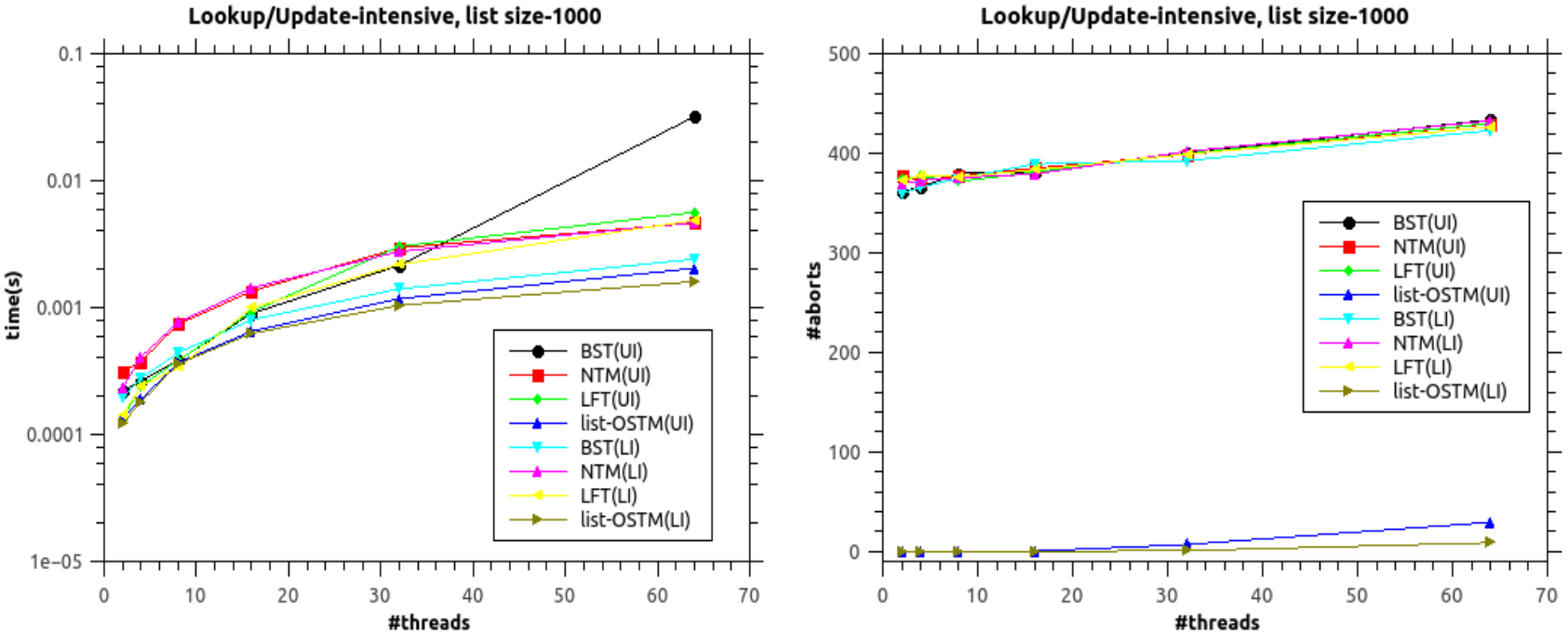

Figure: time vs #threads OR #aborts vs #threads

Content

Conclusion and Future Work

OSTM = Efficiency + Composition + Programmer friendly.

- HT-OSTM based on OSTM is *opaque*.
- HT-OSTM shows speedup of 3 to 6 times better than state of the art ESTM & RWSTM.
- list-OSTM outperforms state of the art LFT, NTM and BST by 30% to 80% across all workloads and scenarios.
- HT-OSTM and list-OSTM are having negligible aborts in comparison to other techniques.

Conclusion and Future Work

OSTM = Efficiency + Composition + Programmer friendly.

- HT-OSTM based on OSTM is *opaque*.
- HT-OSTM shows speedup of 3 to 6 times better than state of the art ESTM & RWSTM.
- list-OSTM outperforms state of the art LFT, NTM and BST by 30% to 80% across all workloads and scenarios.
- HT-OSTM and list-OSTM are having negligible aborts in comparison to other techniques.

Future Work

- The OSTM model can be extended to other data structures like Queue, Stack etc.
- The OSTM model can be extended to **multi-version OSTM (MV-OSTM)** for achieving higher concurrency.

Thank You !

Q&A