भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

An Innovative Approach for Achieving Composability in Concurrent Systems using Multi-Version Object Based STMs

Sandeep Kulkarni[2] **Sweta Kumari**[1] Sathya Peri[1]
Archit Somani[1]

[1]Department of Computer Science Engineering, IIT Hyderabad
[2]Department of Computer Science, Michigan State University

Outline

1. Introduction to STMs
2. Correctness Criteria of STMs
3. Problem with read-write STM
4. Object Based STMs
5. Motivation towards MV-OSTM
6. Correctness of MV-OSTM
7. Conclusion
8. Future Work

Outline

1. Introduction to STMs
2. Correctness Criteria of STMs
3. Problem with read-write STM
4. Object Based STMs
5. Motivation towards MV-OSTM
6. Correctness of MV-OSTM
7. Conclusion
8. Future Work

Introduction to STMs

Software Transactional Memory

What is a transaction?

- Sequence of instructions executing in memory.
- Satisfying ACI

Introduction to STMs

Software Transactional Memory

What is a transaction?

- Sequence of instructions executing in memory.
- Satisfying ACI

What is Software Transactional Memory?

- A parallel programming paradigm
- Avoids concurrency overheads at programmers level
- Execute code optimistically

Introduction to STMs

Software Transactional Memory

What is a transaction?

- Sequence of instructions executing in memory.
- Satisfying ACI

What is Software Transactional Memory?

- A parallel programming paradigm
- Avoids concurrency overheads at programmers level
- Execute code optimistically

Methods of STMs :

- Read
- Write
- TryC

Illustration of STMs methods

Illustration of STMs methods

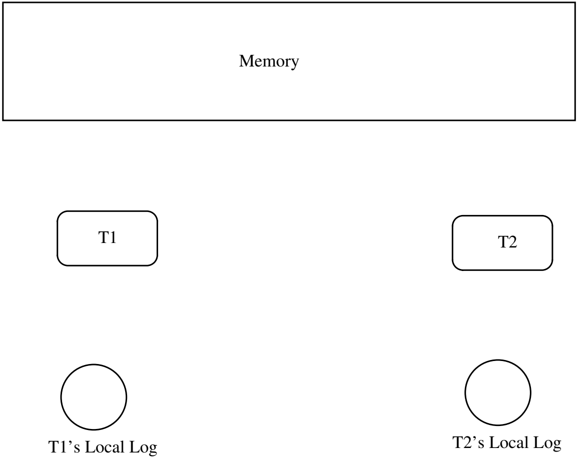


Figure: Working of STM System

Illustration of STMs methods

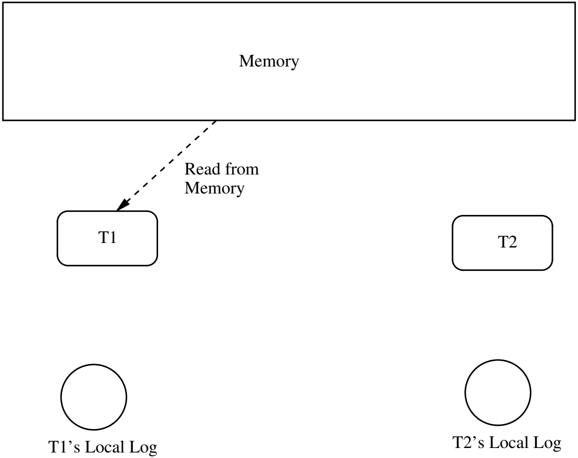


Figure: Working of STM System

Illustration of STMs methods

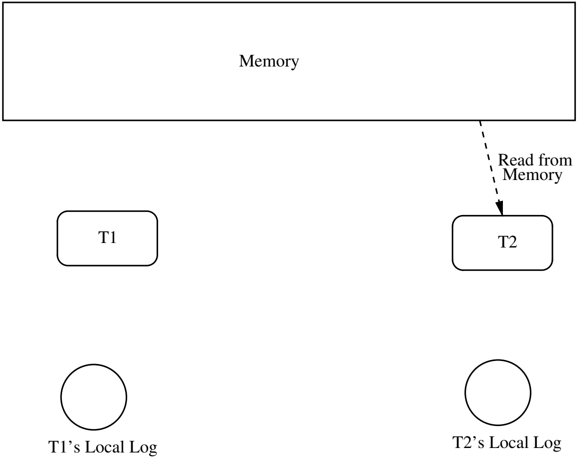


Figure: Working of STM System

Illustration of STMs methods

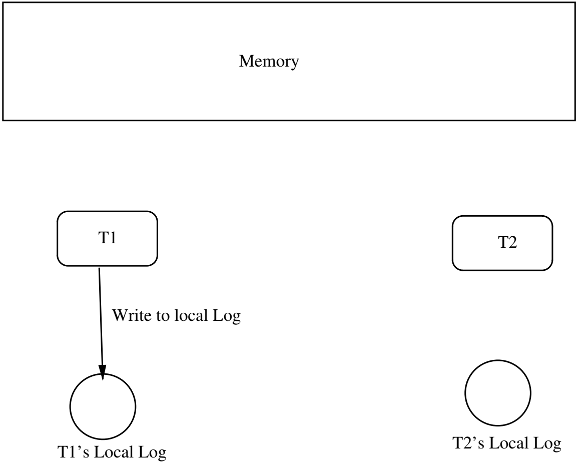


Figure: Working of STM System

Illustration of STMs methods

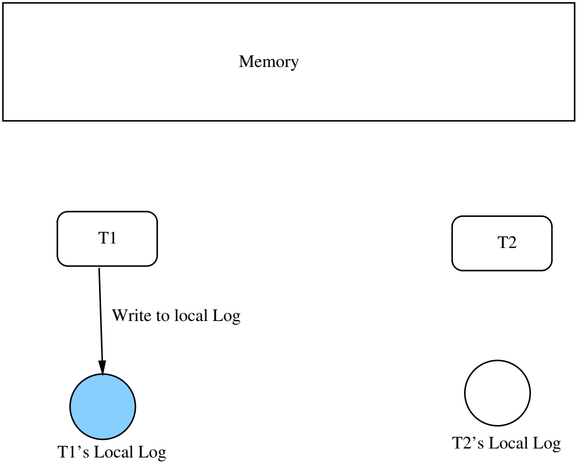


Figure: Working of STM System

Illustration of STMs methods

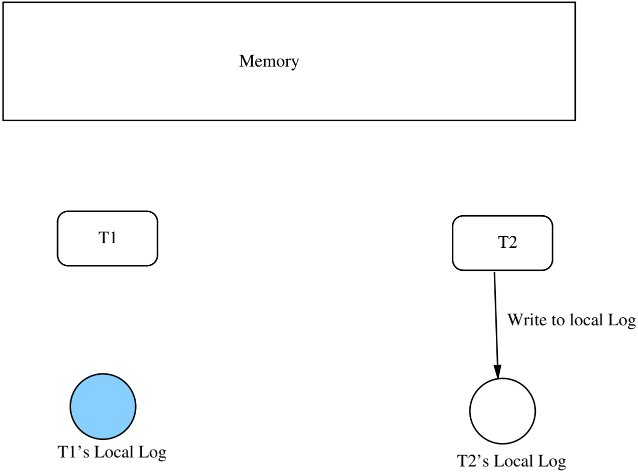


Figure: Working of STM System

Illustration of STMs methods

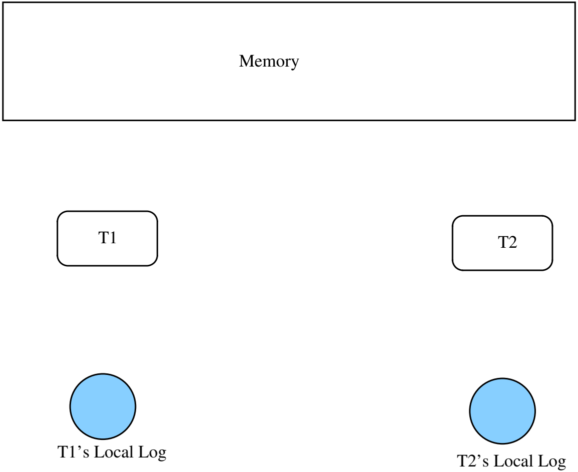


Figure: Working of STM System

Illustration of STMs methods

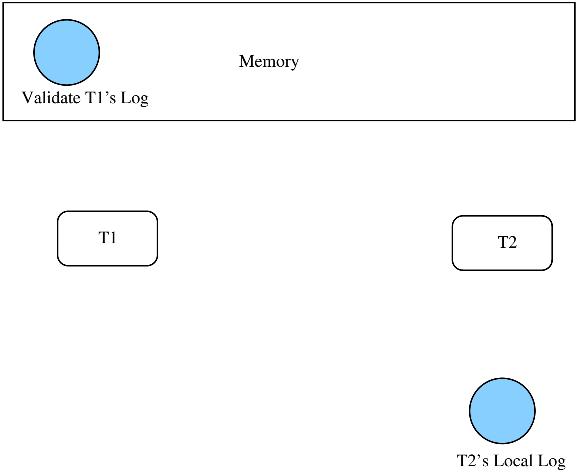


Figure: Working of STM System

Illustration of STMs methods

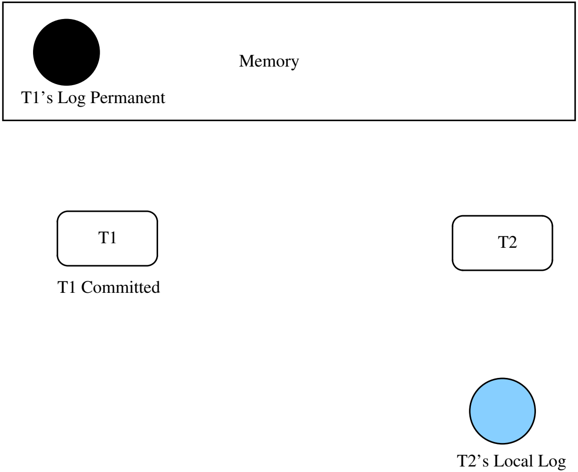


Figure: Working of STM System

Illustration of STMs methods

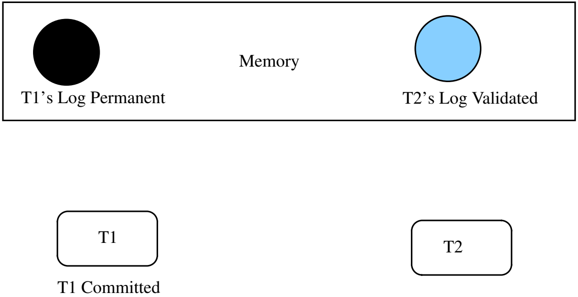


Figure: Working of STM System

Illustration of STMs methods

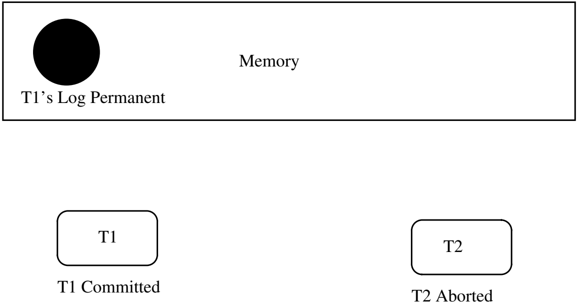


Figure: Working of STM System

Outline

1. Introduction to STMs
2. Correctness Criteria of STMs
3. Problem with read-write STM
4. Object Based STMs
5. Motivation towards MV-OSTM
6. Correctness of MV-OSTM
7. Conclusion
8. Future Work

Correctness of STM System

Correctness of STM System

Correcness criteria for STMs (Opacity)

- 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 are legal in S

Correctness of STM System

Example of opacity

- H: r_1(x,0)w_2(x,10)w_2(y,10)$C_2 r_1$(y,A)A_1

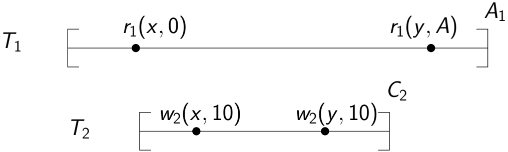


Figure: Opaque History H

Correctness of STM System

Example of opacity

- H: r_1(x,0)w_2(x,10)w_2(y,10)C_2r_1(y,A)A_1

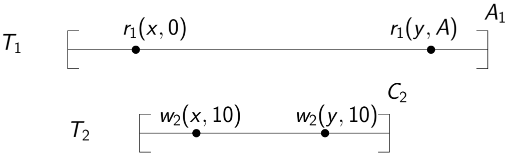


Figure: Opaque History H

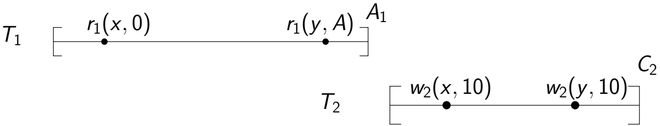


Figure: Equivalent serial history S: T_1, T_2

Outline

1. Introduction to STMs
2. Correctness Criteria of STMs
3. Problem with read-write STM
4. Object Based STMs
5. Motivation towards MV-OSTM
6. Correctness of MV-OSTM
7. Conclusion
8. Future Work

Problem with read-write STM

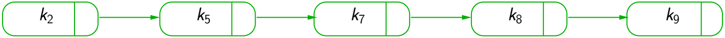


Figure: A sample concurrent object

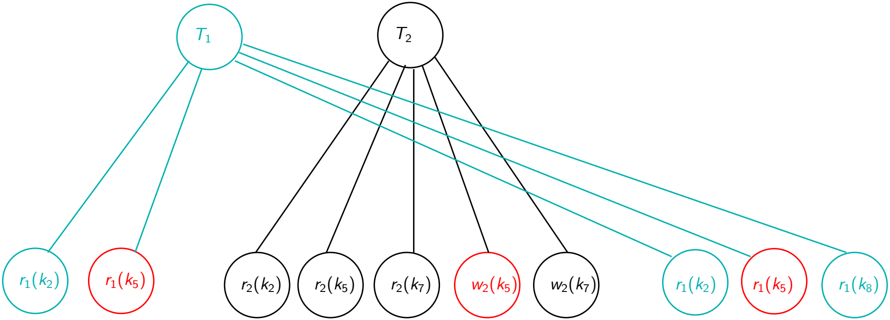


Figure: Tree Structure : conflicts are $(r_1(k_5), w_2(k_5))$ and $(w_2(k_5), r_1(k_5)$

Problem at read-write level

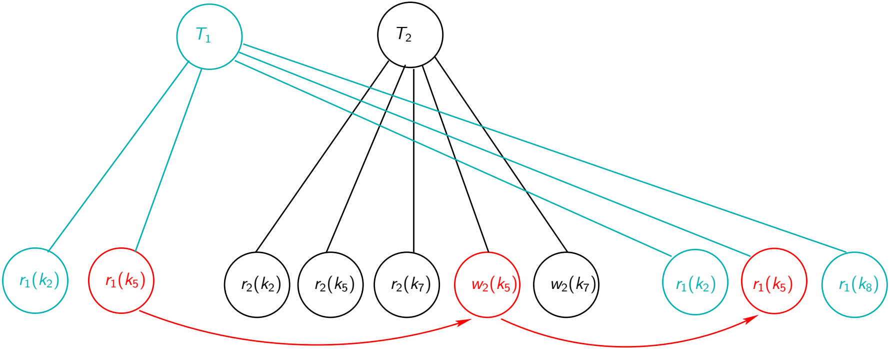


Figure: Tree Structure

Problem at read-write level

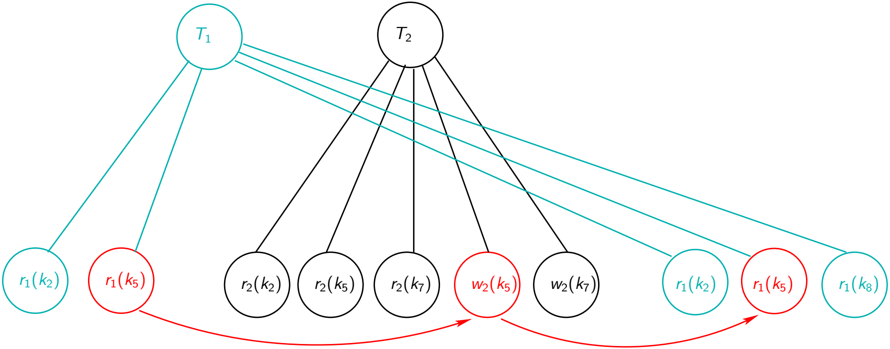


Figure: Tree Structure

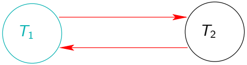


Figure: Cycle (Not Serial)

Outline

1. Introduction to STMs
2. Correctness Criteria of STMs
3. Problem with read-write STM
4. Object Based STMs
5. Motivation towards MV-OSTM
6. Correctness of MV-OSTM
7. Conclusion
8. Future Work

OSTM

Introduction

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

OSTM

Introduction

- **Object-based STMs (OSTM)** operate on higher level objects rather than primitive read & writes which act upon memory locations.
- OSTM model can adapted:
    - OSTM for stacks may export *t_push*, *t_pop* & *t_peek*.
    - OSTM for sets may export *t_begin(), t_insert(), t_del(), t_lookup() and tryC()*.

OSTM

Execution at layer-1

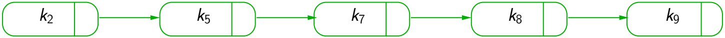


Figure: A sample representing a OSTM object

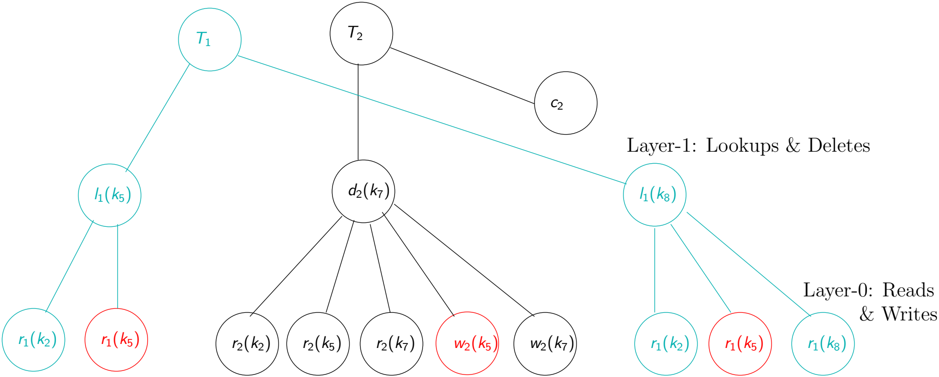


Figure: Tree Structure : no conflict at Layer-1

OSTM

Execution at layer-1

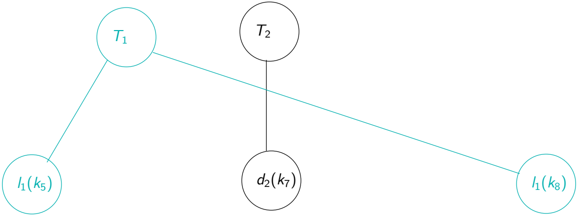


Figure: Pruned Tree

OSTM

Execution at layer-1

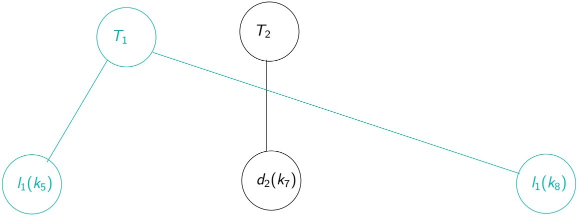


Figure: Pruned Tree

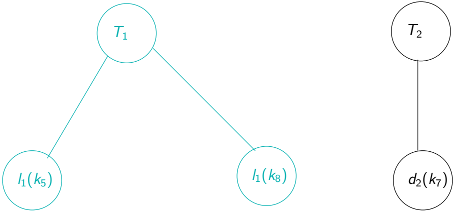


Figure: Sequential Schedule

OSTM

Execution at layer-1

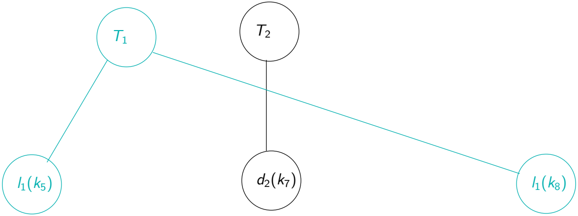


Figure: Pruned Tree

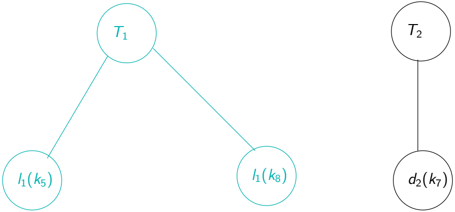


Figure: Sequential Schedule

Figure: Serial History

OSTM

Problem with OSTM

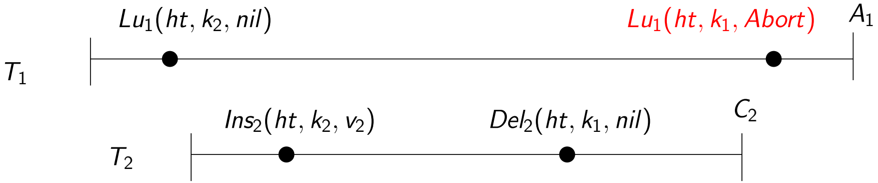


Figure: Single version OSTM

Outline

1. Introduction to STMs
2. Correctness Criteria of STMs
3. Problem with read-write STM
4. Object Based STMs
5. Motivation towards MV-OSTM
6. Correctness of MV-OSTM
7. Conclusion
8. Future Work

Proposed Algorithm : MV-OSTM

Advantages of multi-version over single version OSTM

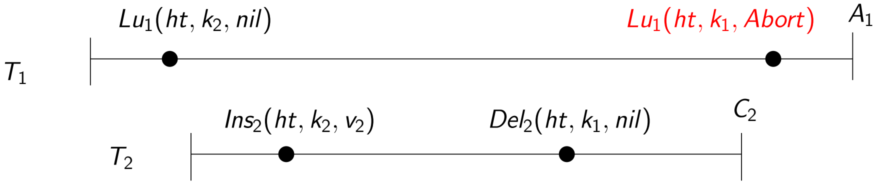


Figure: Single version OSTM

Proposed Algorithm : MV-OSTM

Advantages of multi-version over single version OSTM

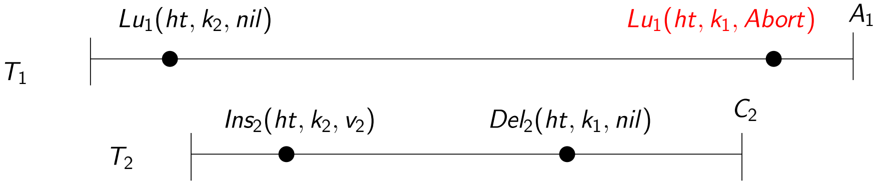


Figure: Single version OSTM

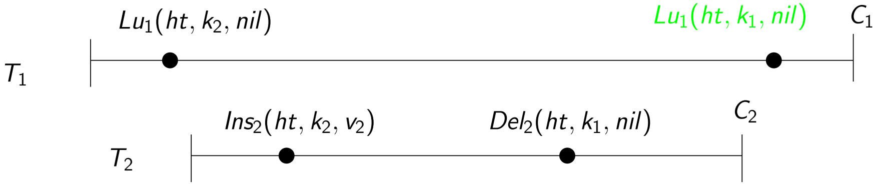
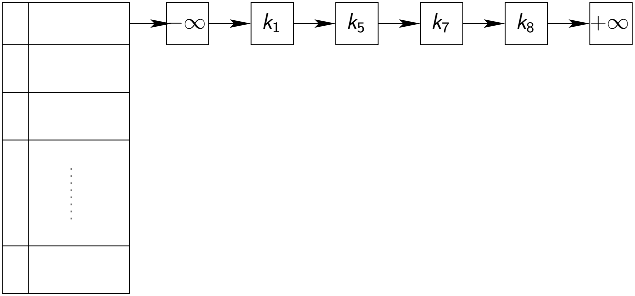


Figure: Multi-version OSTM (MV-OSTM) : (T_1, T_2)

Proposed Algorithm : MV-OSTM

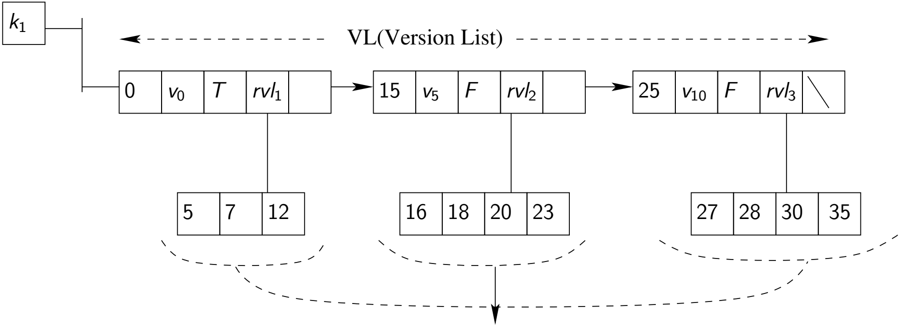
Illustration of data structure



a) Underlying DS

Proposed Algorithm : MV-OSTM

Illustration of data structure cont'd..



RVL(Return Value List)

b) DS for maintaining Versions

Proposed Algorithm : MV-OSTM

Lookup method

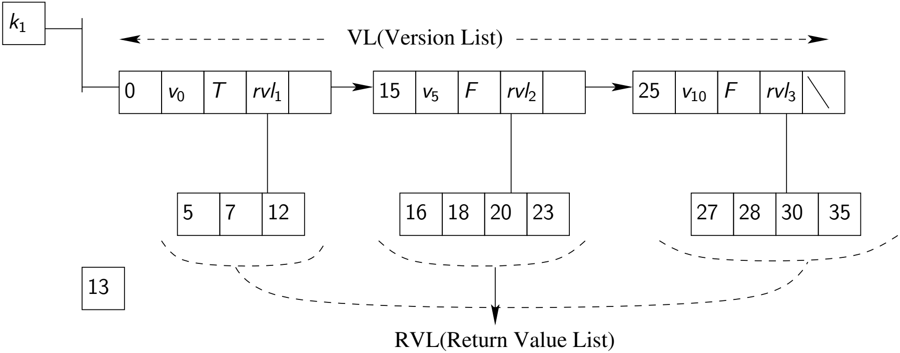


Figure: Lookup on key k_1 by T_{13}

Proposed Algorithm : MV-OSTM

Lookup method cont'd..

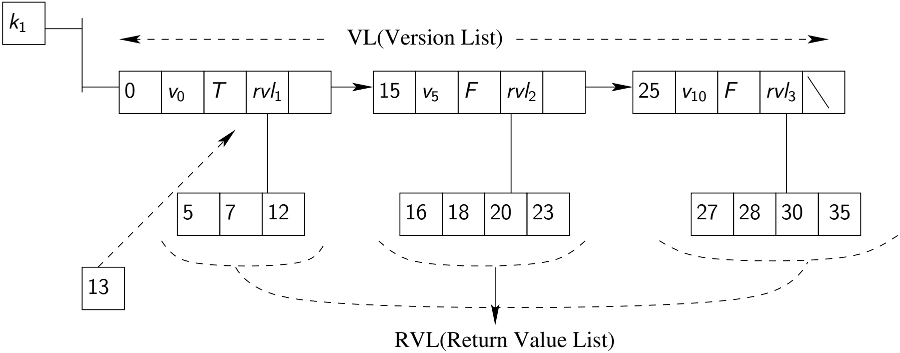


Figure: T_{13} searching appropriate place in version list of k_1

Proposed Algorithm : MV-OSTM

Lookup method cont'd..

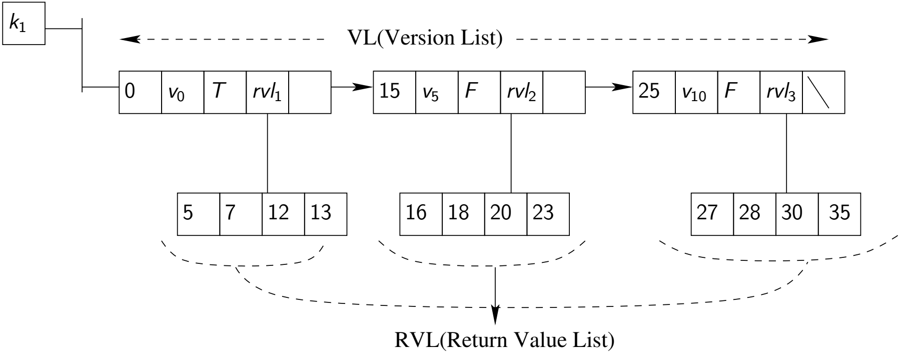


Figure: T_{13} successfully added into rvl_1

Proposed Algorithm : MV-OSTM

tryC : Insert method

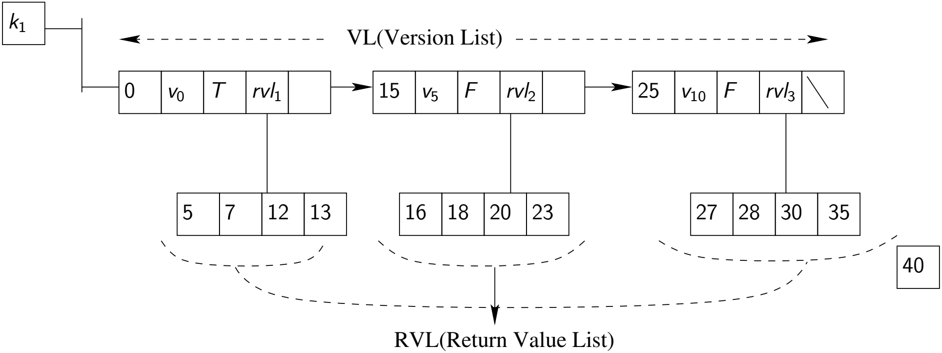


Figure: Insert a version of key k_1 by T_{40}

Proposed Algorithm : MV-OSTM

tryC : Insert method cont'd..

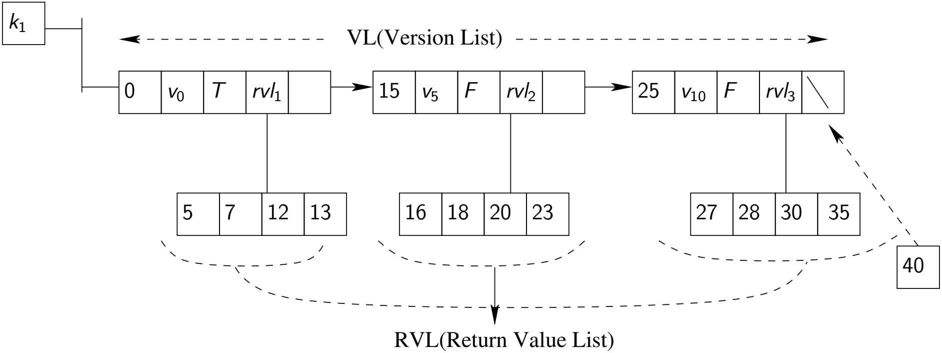


Figure: T_{40} searching appropriate place in version list of k_1

Proposed Algorithm : MV-OSTM

tryC : Insert method cont'd..

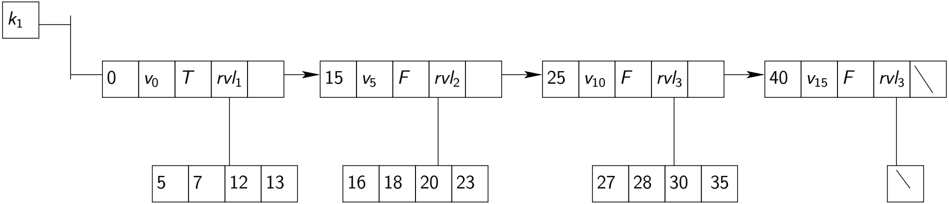


Figure: T_{40} successfully created a new version of k_1

Proposed Algorithm : MV-OSTM

tryC : Insert method cont'd..

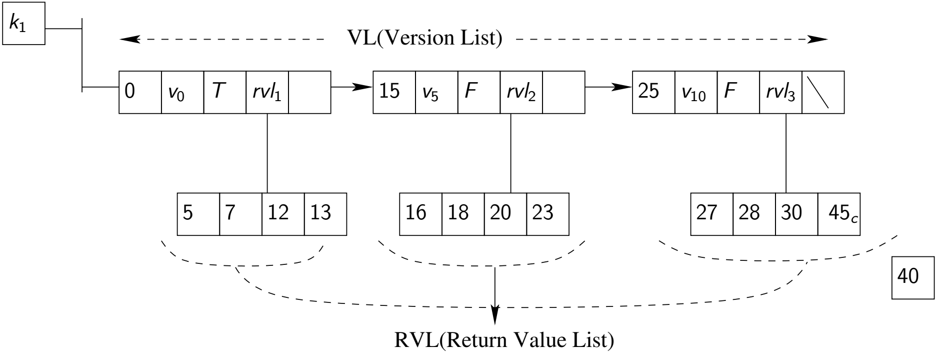


Figure: Insert a version of key k_1 by T_{40}

Proposed Algorithm : MV-OSTM

tryC : Insert method cont'd..

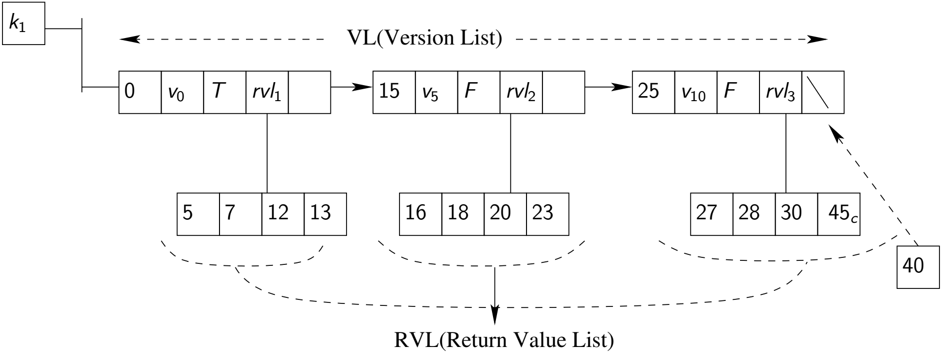


Figure: T_{40} searching appropriate place in version list of k_1

Proposed Algorithm : MV-OSTM

tryC : Insert method cont'd..

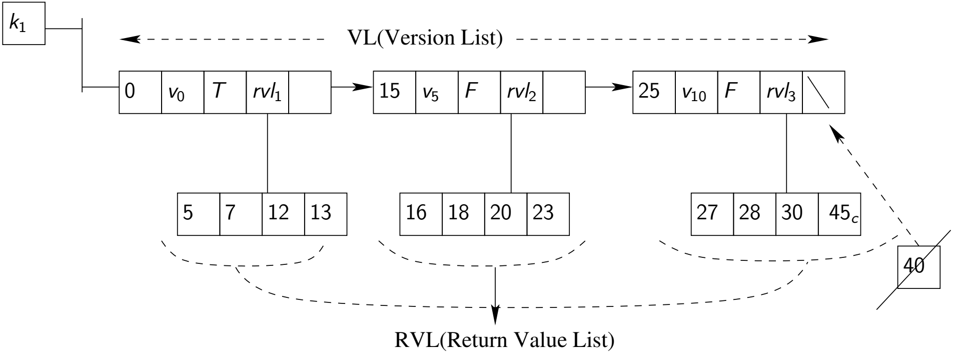


Figure: Abort T_{40} : T_{45} committed before T_{40}

Outline

1. Introduction to STMs
2. Correctness Criteria of STMs
3. Problem with read-write STM
4. Object Based STMs
5. Motivation towards MV-OSTM
6. Correctness of MV-OSTM
7. Conclusion
8. Future Work

Correctness of MV-OSTM

Theorem

Any history H generated by MV-OSTM algorithm with a given version order $\ll$, if $OPG(H, \ll)$ is acyclic, then H is opaque.

Outline

1. Introduction to STMs
2. Correctness Criteria of STMs
3. Problem with read-write STM
4. Object Based STMs
5. Motivation towards MV-OSTM
6. Correctness of MV-OSTM
7. Conclusion
8. Future Work

Conclusion

- MV-OSTM is *opaque*.

Conclusion

- MV-OSTM is *opaque*.
- We have proposed a new STM as MV-OSTM which providing the greater concurrency in terms of the number of aborts with the help of multiple versions and composability.

Conclusion

- MV-OSTM is *opaque*.
- We have proposed a new STM as MV-OSTM which providing the greater concurrency in terms of the number of aborts with the help of multiple versions and composability.
- Lookup operation always succeeds.

Conclusion

- MV-OSTM is *opaque*.
- We have proposed a new STM as MV-OSTM which providing the greater concurrency in terms of the number of aborts with the help of multiple versions and composability.
- Lookup operation always succeeds.
- Delete operation is logically deletes, in that sense it's lazy.

Conclusion

- MV-OSTM is *opaque*.
- We have proposed a new STM as MV-OSTM which providing the greater concurrency in terms of the number of aborts with the help of multiple versions and composability.
- Lookup operation always succeeds.
- Delete operation is logically deletes, in that sense it's lazy.
- Transactions are *composable [Harris et.al, 2005]*, *[Ziv et.al, 2015]*.

Outline

1. Introduction to STMs
2. Correctness Criteria of STMs
3. Problem with read-write STM
4. Object Based STMs
5. Motivation towards MV-OSTM
6. Correctness of MV-OSTM
7. Conclusion
8. Future Work

Progress

Future Work

- Garbage Collection.

Progress

Future Work

- Garbage Collection.
- We will extend it for K-version MV-OSTM.

Progress

Future Work

- Garbage Collection.
- We will extend it for K-version MV-OSTM.
- We will implement our proposed protocol and compare the performance with existing Object-Based STMs *[Hassan et.al, 2014]*.

Future Work

- Garbage Collection.
- We will extend it for K-version MV-OSTM.
- We will implement our proposed protocol and compare the performance with existing Object-Based STMs *[Hassan et.al, 2014]*.
- Nesting : open *[Yang et.al, 2007]* and close.

References

1. Harris, Tim and Marlow, Simon and Peyton-Jones, Simon and Herlihy, Maurice. Composable memory transactions, Proceedings of the tenth ACM SIGPLAN symposium on Principles and practice of parallel programming, 2005
2. Gerhard Weikum and Gottfried Vossen. 2001. Transactional Information Systems: Theory, Algorithms, and the Practice of Concurrency Control and Recovery. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA
3. Ziv, Ofri and Aiken, Alex and Golan-Gueta, Guy and Ramalingam, G and Sagiv, Mooly, Composing concurrency control, Proceedings of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation, 2015
4. Kuznetsov, Petr and Peri, Sathya, Non-interference and Local Correctness in Transactional Memory, ICDCN, 2014
5. Hassan, Ahmed and Palmieri, Roberto and Ravindran, Binoy, Optimistic Transactional Boosting, Proceedings of the 19th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, PPoPP '14
6. Priyanka Kumar, Sathya Peri, and K. Vidyasankar. A TimeStamp Based Multi-version STM Algorithm. In ICDCN, pages 212–226, 2014
7. Ni, Yang and Menon, Vijay S. and Adl-Tabatabai, Ali-Reza and Hosking, Antony L. and Hudson, Richard L. and Moss, J. Eliot B. and Saha, Bratin and Shpeisman, Tatiana, Open Nesting in Software Transactional Memory, PPoPP '07

Thank You!

Any Questions?