Verifying Concurrent Multicopy Search Structures

Nisarg Patel[1] Siddharth Krishna[2] Dennis Shasha[1] Thomas Wies[1]

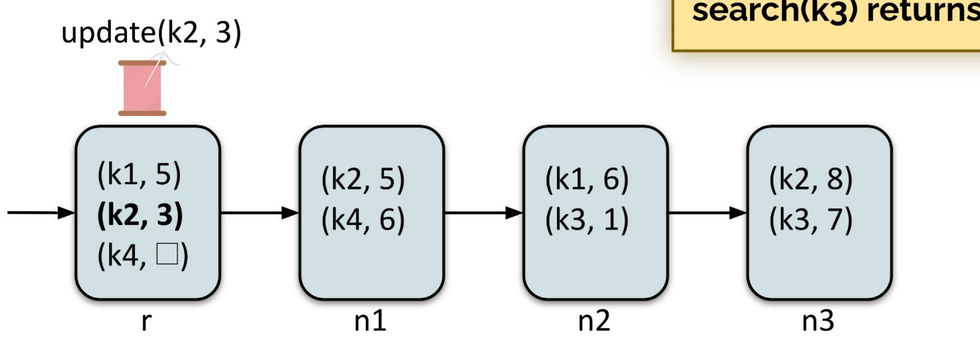
1: NYU COURANT 2: Microsoft

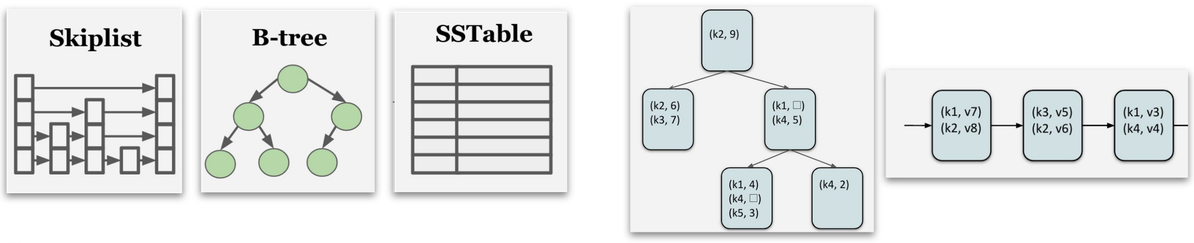
Multicopy Search Structures

Optimized for write-heavy workload

delete(k) ~ upsert(k, □)

search(k3) returns (k3, 1)





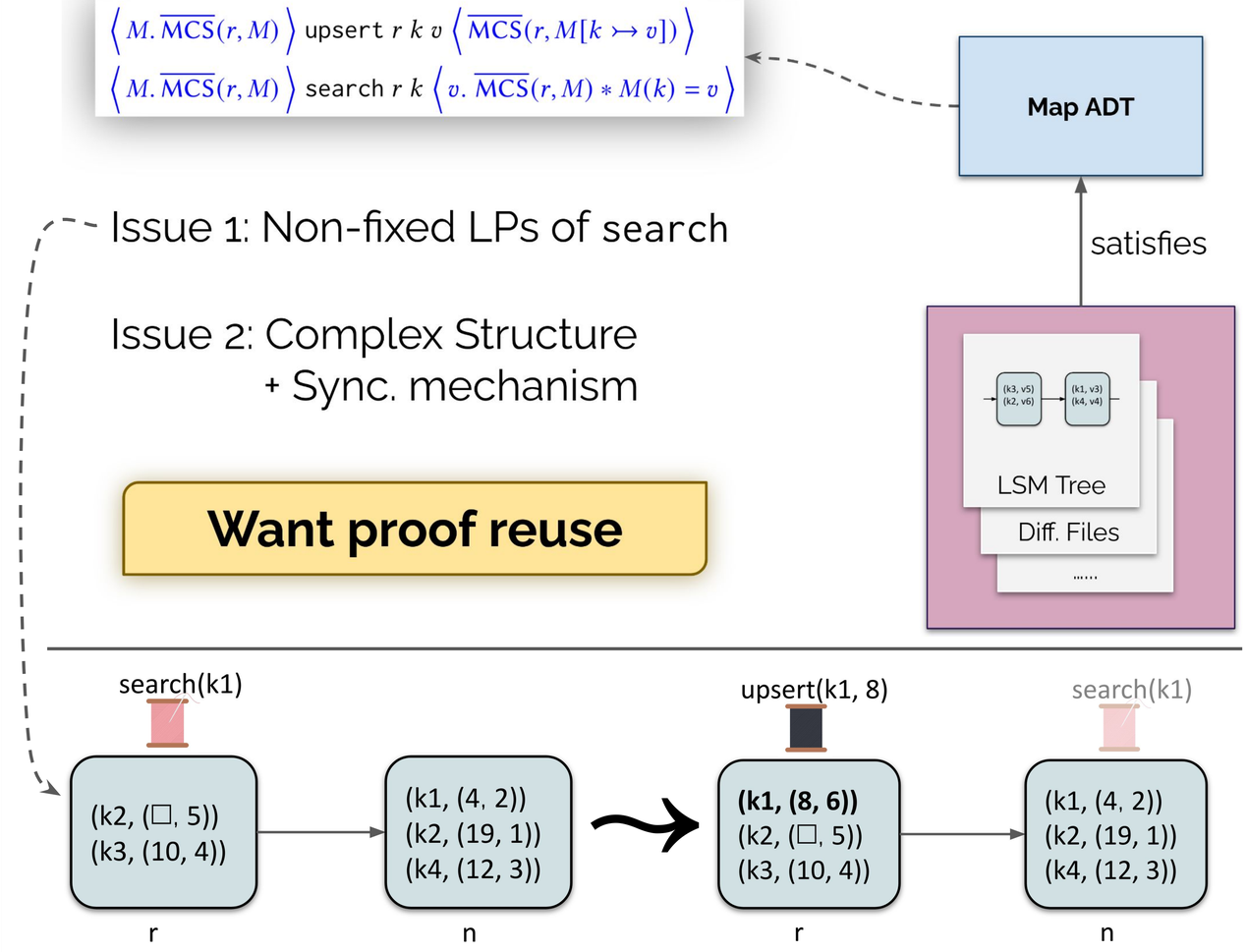
Examples:

Log Structured Merge (LSM) Tree,
Differential Files, etc.

Proving correctness is difficult

$$\left\langle M.\,\overline{\mathrm{MCS}}(r, M) \right\rangle \ \texttt{upsert}\ r\ k\ v \ \left\langle \overline{\mathrm{MCS}}(r, M[k \rightarrowtail v]) \right\rangle$$

$$\left\langle M.\,\overline{\mathrm{MCS}}(r, M) \right\rangle \ \texttt{search}\ r\ k \ \left\langle v.\ \overline{\mathrm{MCS}}(r, M) * M(k) = v \right\rangle$$



Issue 1: Non-fixed LPs of `search`

Issue 2: Complex Structure
+ Sync. mechanism

Want proof reuse

Solution to Issue 1: Search Recency

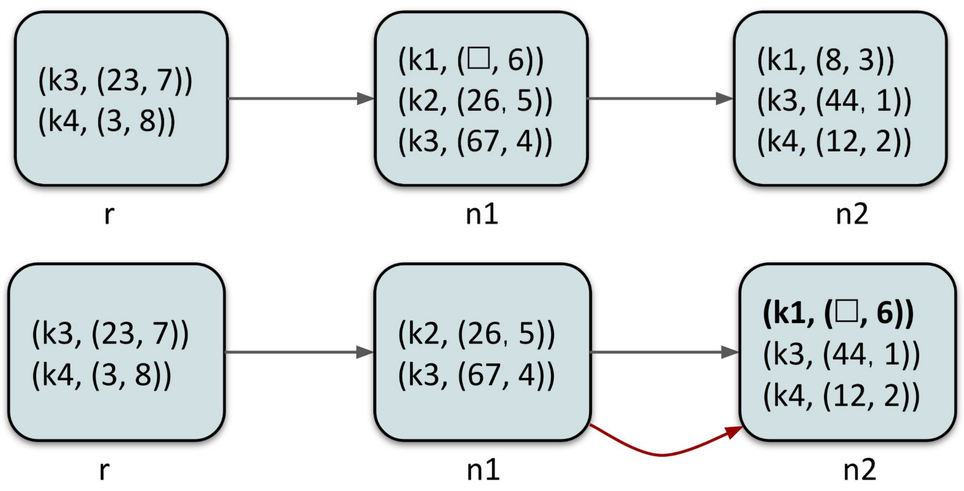
Let `(v0, t0)` = most recent copy of `k` when `search` begins.

Then, `search` either returns:
1) `(v0, t0)` or
2) some `(v, t)` such that `t > t0`.

Linearize at the beginning

Require helping

Invariant: **"first copy reachable from the root is the most recent"**



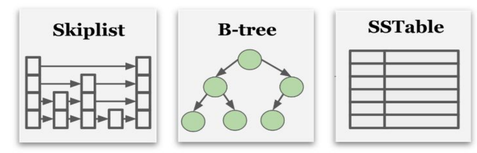
Solution to Issue 2: Template Algorithms

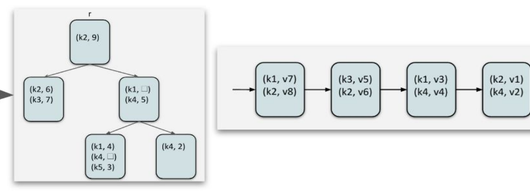
LSM DAG Template

```
let search r k = traverse r k

let rec traverse n k =
    lockNode n;
    match inContents n k with
    | Some v -> unlockNode n; v
    | None ->
      match findNext n k with
      | Some n' ->
        unlockNode n;
        traverse n' k
      | None -> unlockNode n; □
```

Helper function



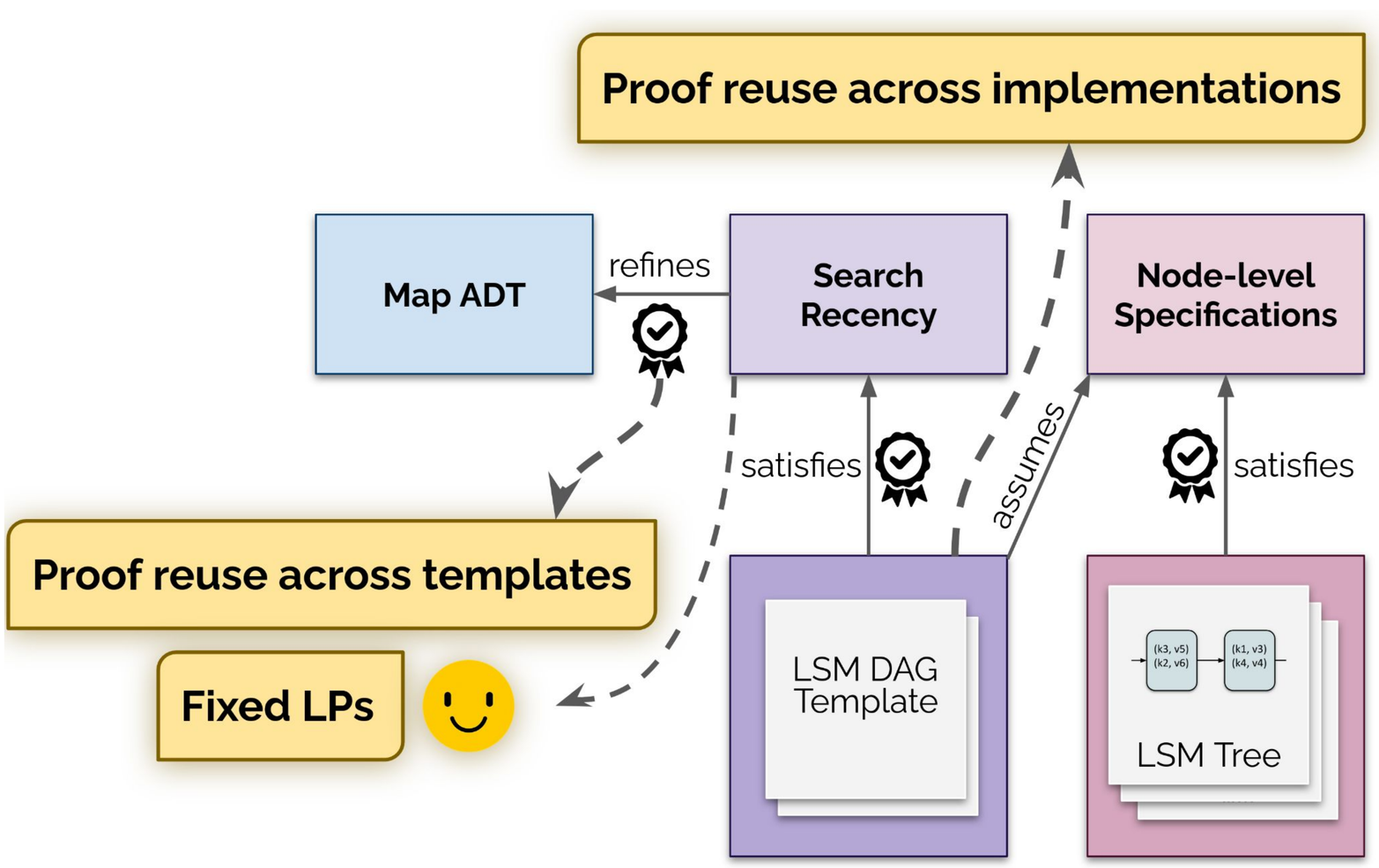


LSM DAG Template

Differential File Template

.....
.....

Overview: Putting it all together



Templates (Iris/Coq)

Module	Code	Proof	Total	Time
Flow Library	0	3757	3757	41
Lock Implementation	10	333	343	10
Client-level Spec	2	792	794	31
DF Template	26	934	960	68
LSM DAG Template	46	3587	3633	307
Total	**84**	**9403**	**9487**	**457**

Implementations (GRASShopper)

Module	Code	Proof	Total	Time
Array Library	191	440	631	10
LSM Implementation	209	222	431	25
Total	**400**	**662**	**1062**	**35**

Simpler and reusable formal proof of real-world data structures!