

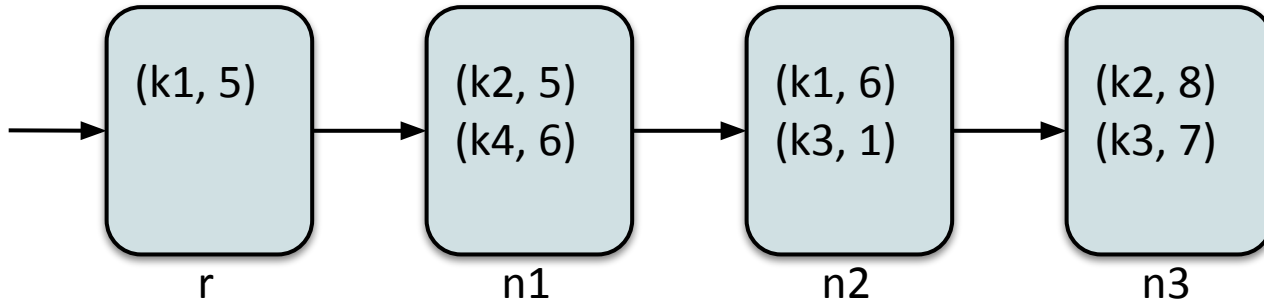
Verifying Concurrent Multicopy Search Structures

Nisarg Patel

Joint work with Siddharth Krishna, Dennis Shasha and Thomas Wies.

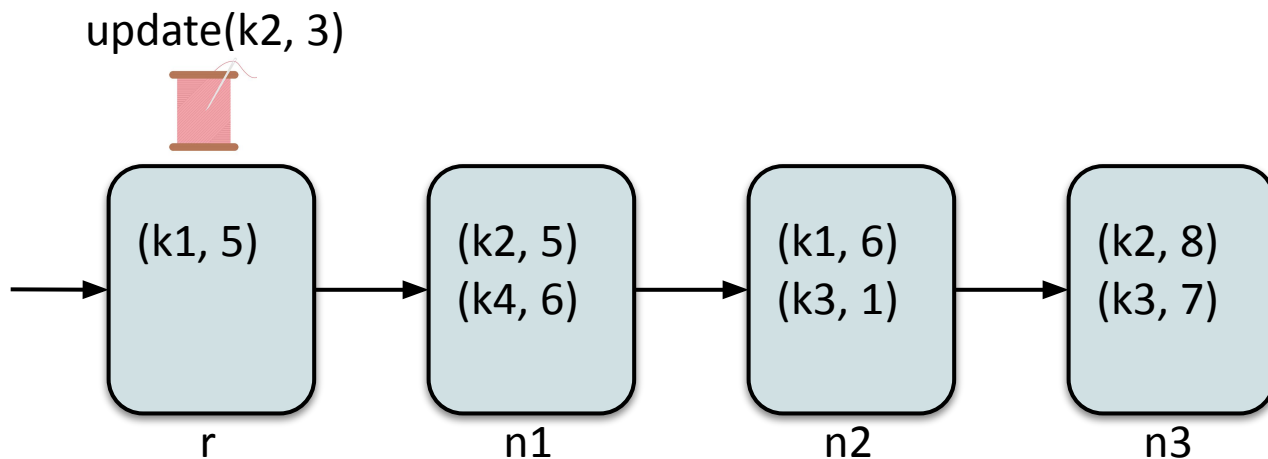
Multicopy Search Structures

Optimized for write-heavy workload



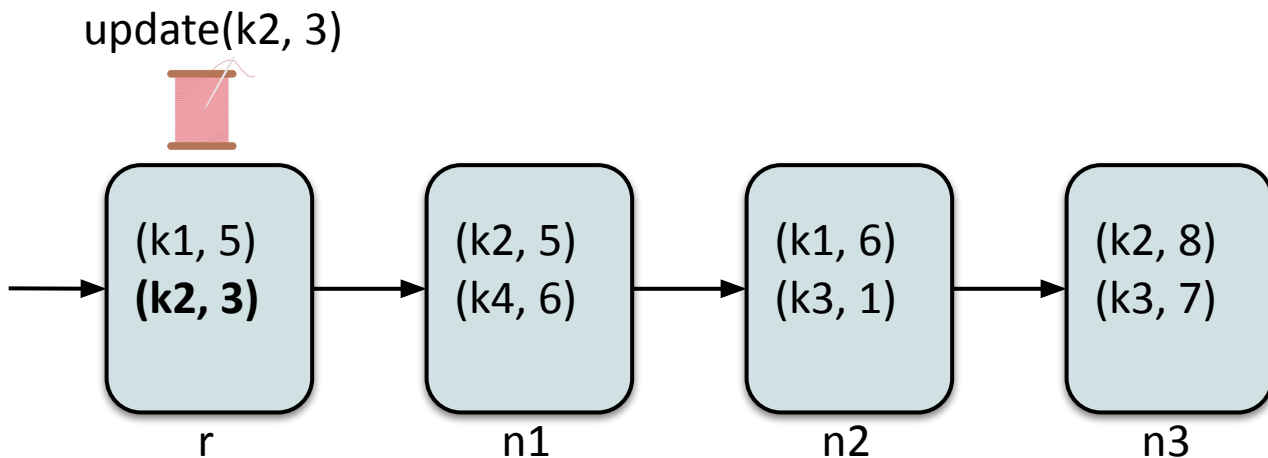
Multicopy Search Structures

Optimized for write-heavy workload



Multicopy Search Structures

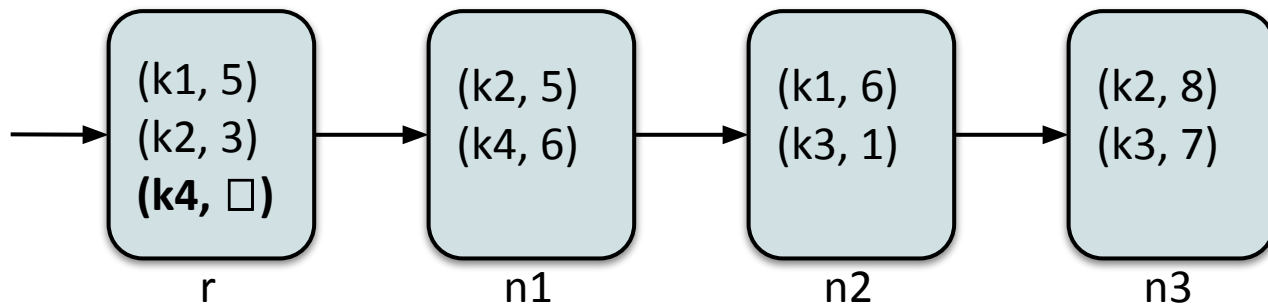
Optimized for write-heavy workload



Multicopy Search Structures

Optimized for write-heavy workload

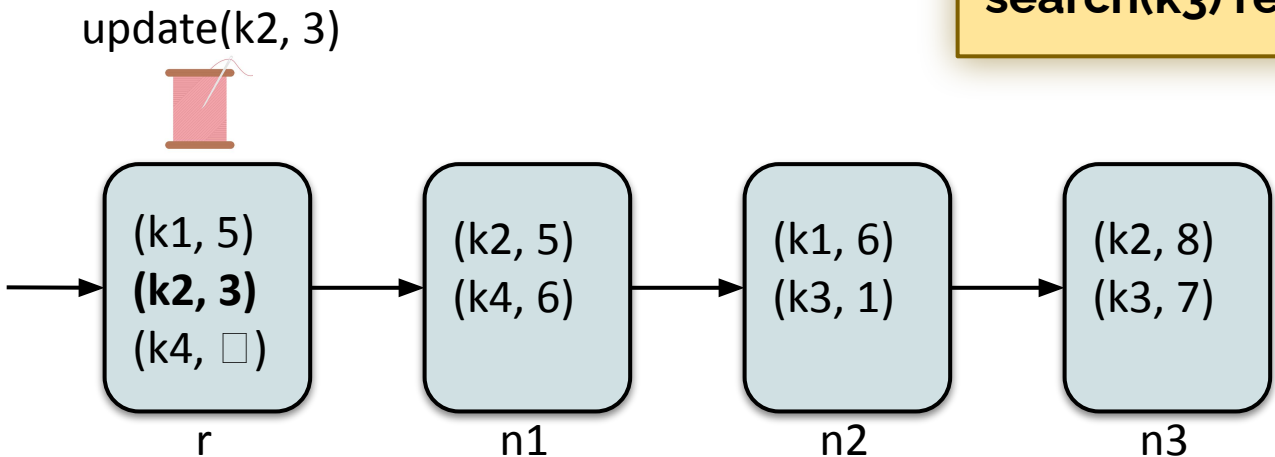
`delete(k) ~ upsert(k, □)`



Optimized for write-heavy workload

delete(k) ~ upsert(k, □)

search(k3) returns (k3, 1)

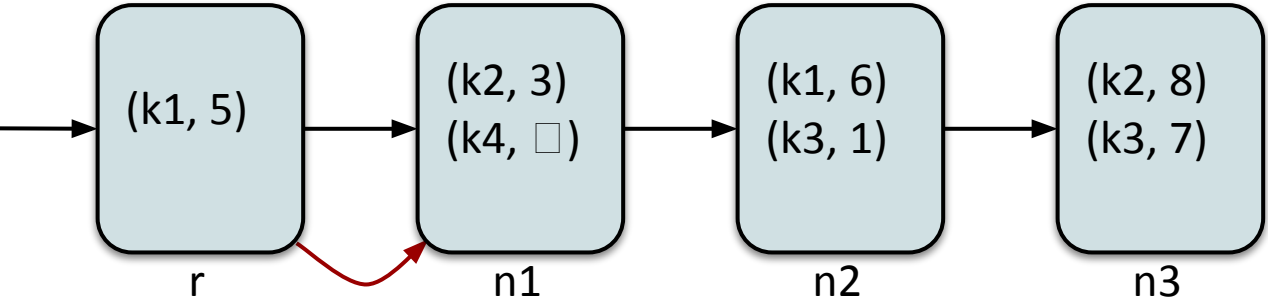


Multicopy Search Structures

Optimized for write-heavy workload

delete(k) ~ upsert(k, □)

search(k3) returns (k3, 1)



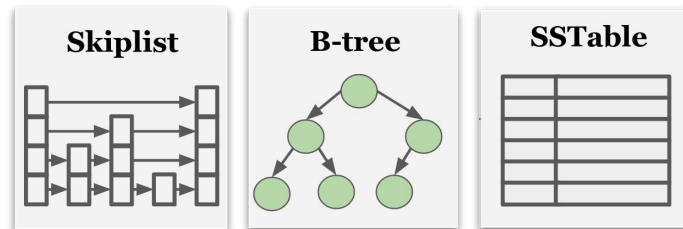
maintenance

Goal

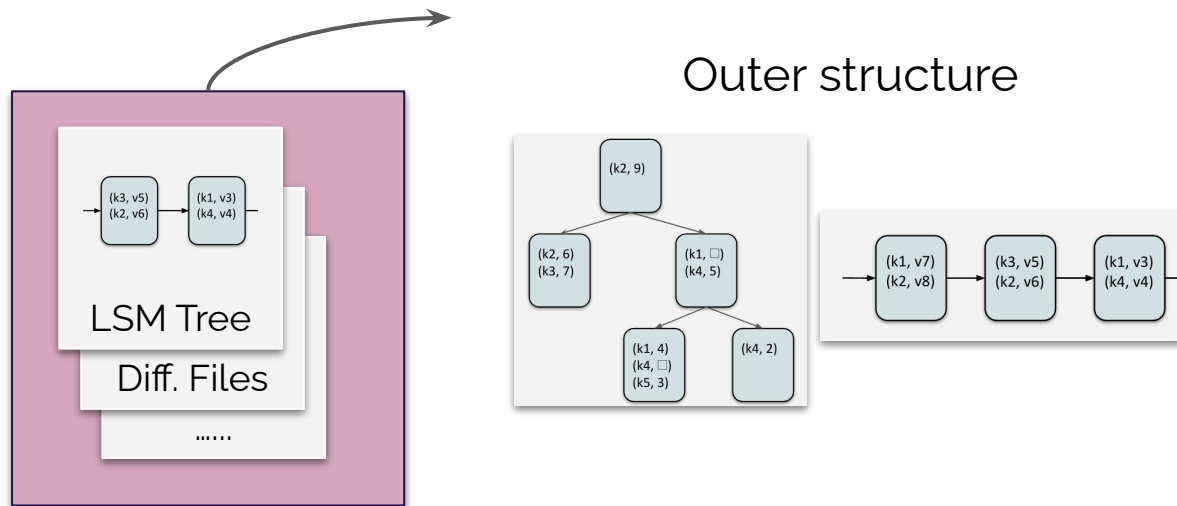


Goal

Inner structure

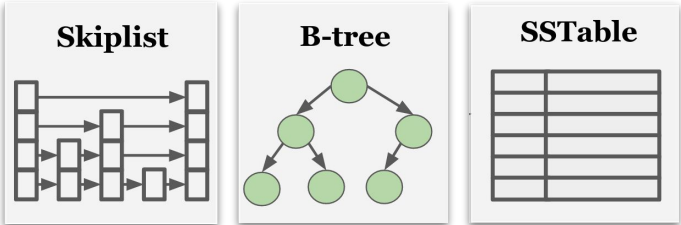


Outer structure

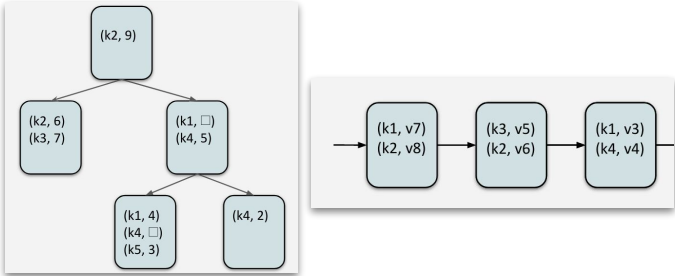


Goal

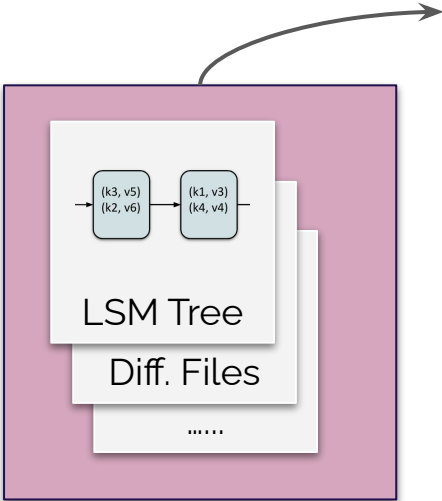
Inner structure



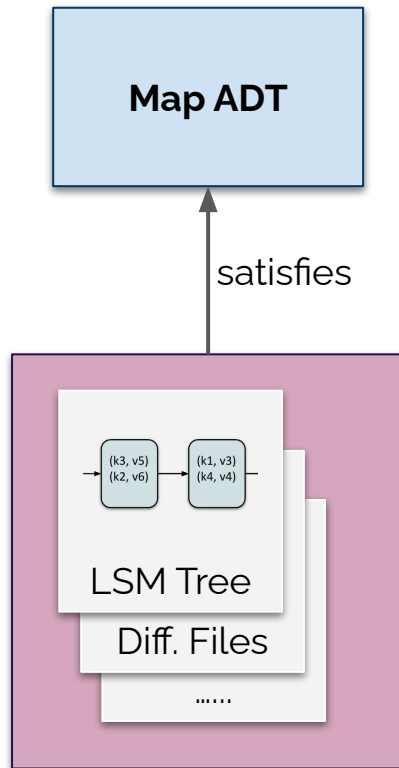
Outer structure



Want proof reuse

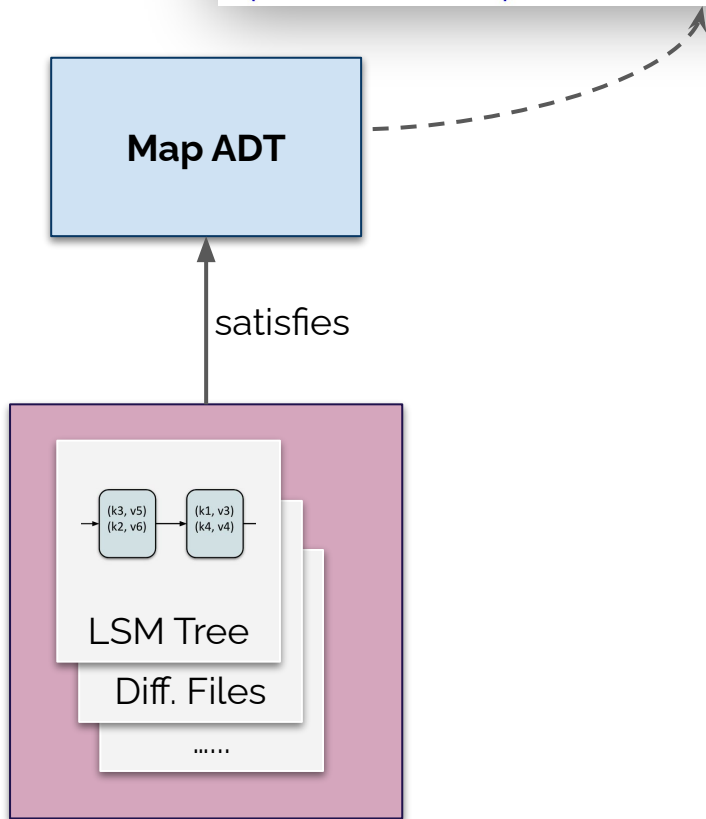


Goal



Goal

$$\langle M. \overline{\text{MCS}}(r, M) \rangle \text{ upsert } r \ k \ v \ \langle \overline{\text{MCS}}(r, M[k \mapsto v]) \rangle$$
$$\langle M. \overline{\text{MCS}}(r, M) \rangle \text{ search } r \ k \ \langle v. \overline{\text{MCS}}(r, M) * M(k) = v \rangle$$

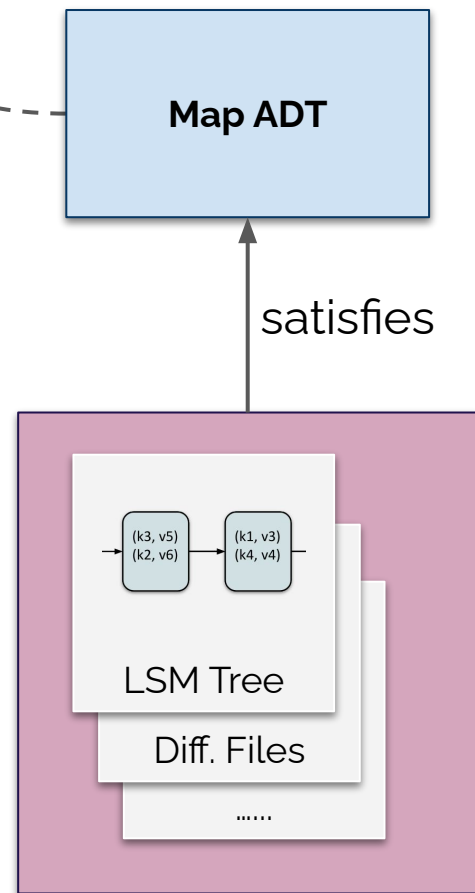


$\langle M. \overline{MCS}(r, M) \rangle$ upsert $r k v \langle \overline{MCS}(r, M[k \mapsto v]) \rangle$
 $\langle M. \overline{MCS}(r, M) \rangle$ search $r k \langle v. \overline{MCS}(r, M) * M(k) = v \rangle$

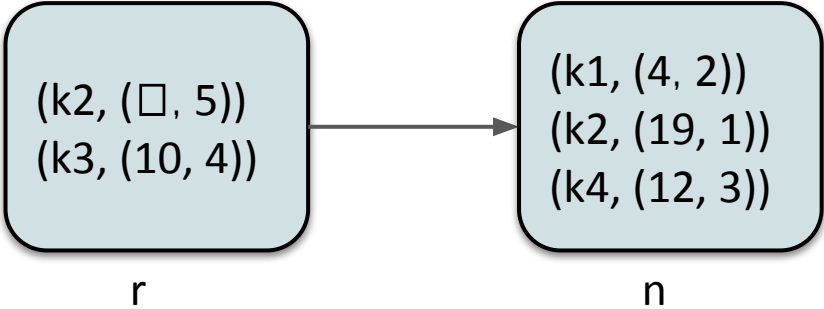
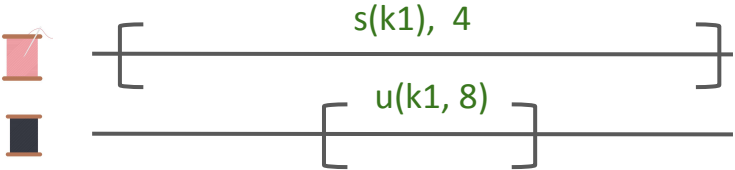
Issue 1: Non-fixed LPs of search

Issue 2: Complex Structure
+ Sync. mechanism

Want proof reuse

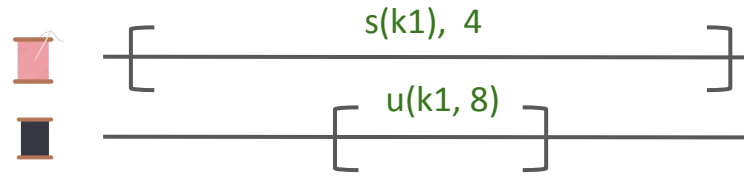


Problem Execution

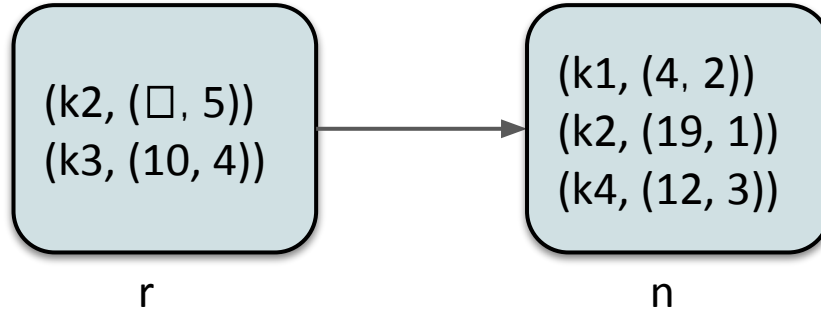


$$\langle M. \overline{MCS}(r, M) \rangle \text{ search } r \text{ } k \langle v. \overline{MCS}(r, M) * M(k) = v \rangle$$

Problem Execution

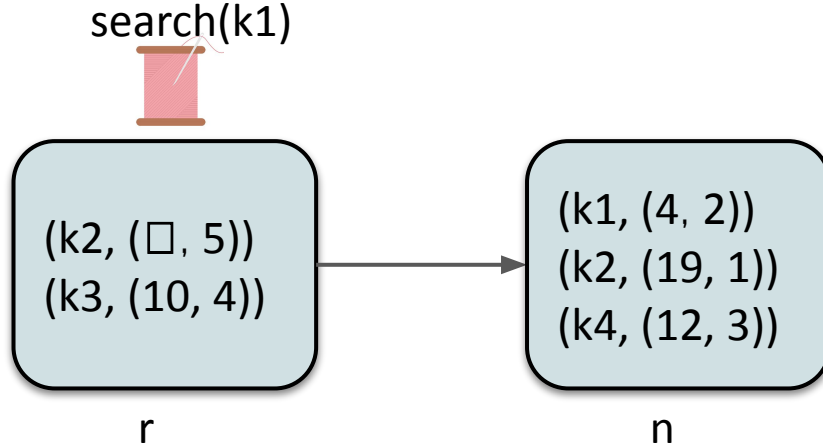
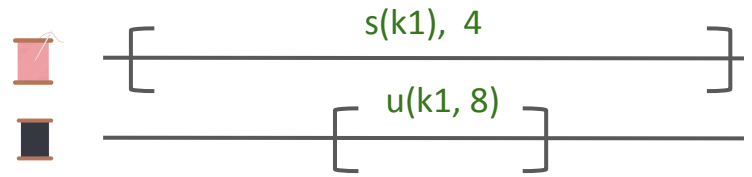


(key, (value, timestamp))



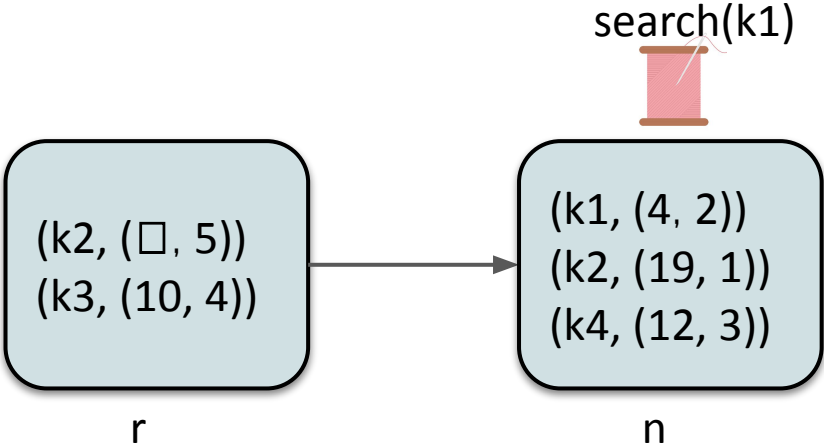
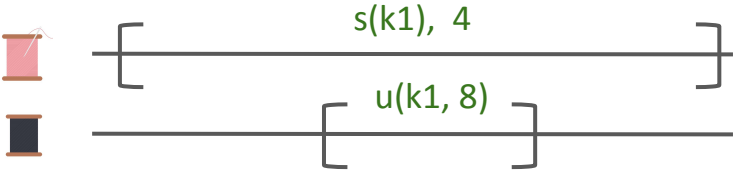
$\langle M. \overline{MCS}(r, M) \rangle$ search r k $\langle v. \overline{MCS}(r, M) * M(k) = v \rangle$

Problem Execution



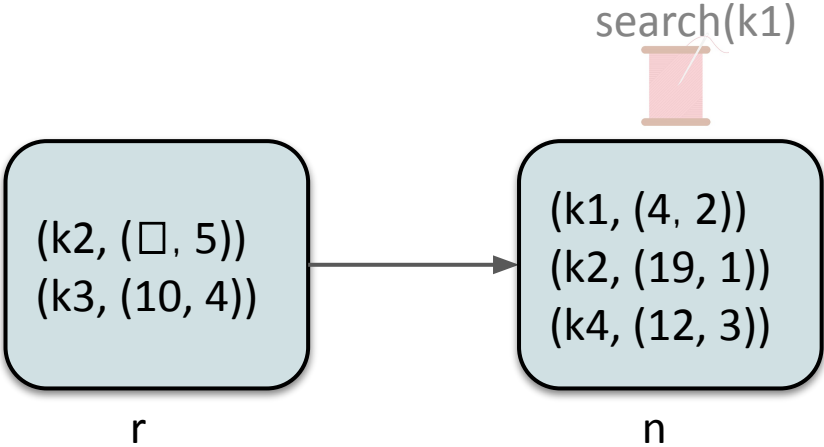
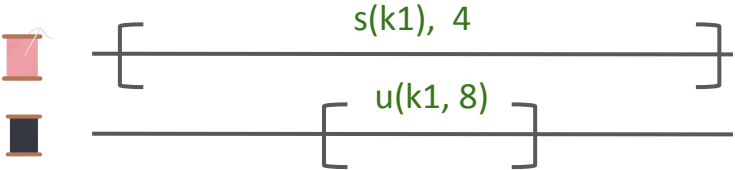
$$\langle M. \overline{\text{MCS}}(r, M) \rangle \text{ search } r \text{ } k \langle v. \overline{\text{MCS}}(r, M) * M(k) = v \rangle$$

Problem Execution



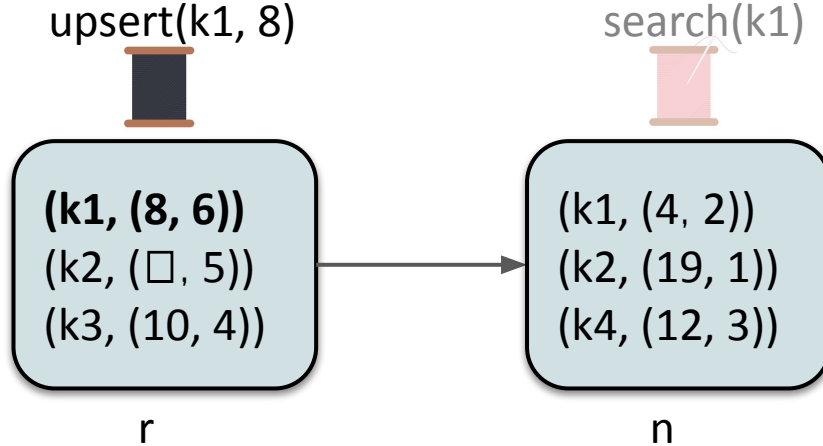
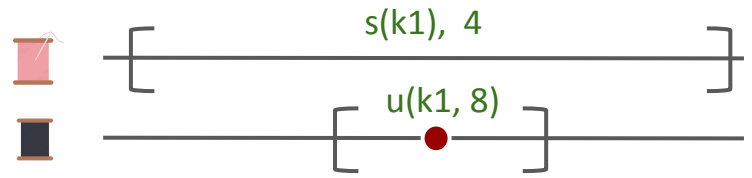
$$\langle M. \overline{\text{MCS}}(r, M) \rangle \text{ search } r \text{ } k \langle v. \overline{\text{MCS}}(r, M) * M(k) = v \rangle$$

Problem Execution



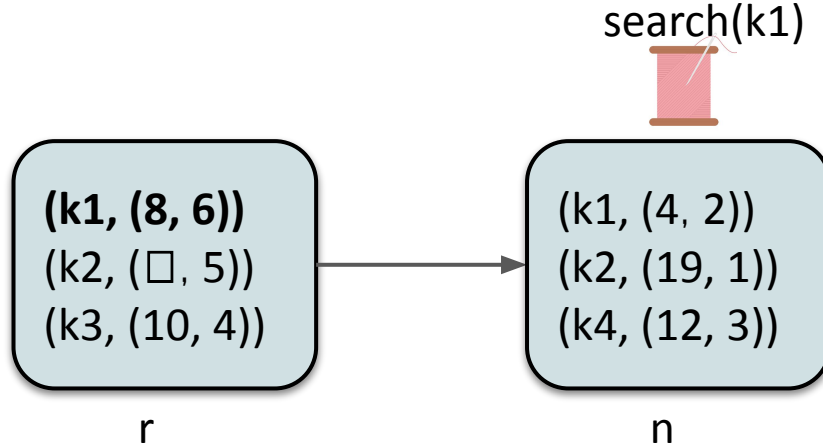
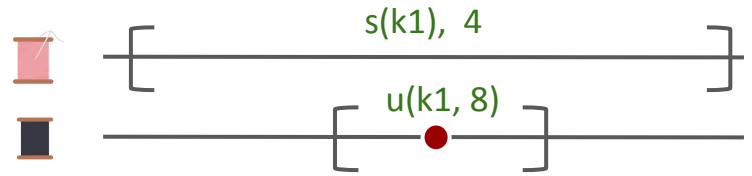
$$\langle M. \overline{MCS}(r, M) \rangle \text{ search } r \text{ } k \langle v. \overline{MCS}(r, M) * M(k) = v \rangle$$

Problem Execution



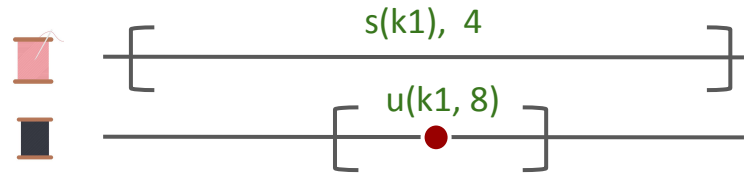
$$\langle M. \overline{MCS}(r, M) \rangle \text{ search } r \ k \langle v. \overline{MCS}(r, M) * M(k) = v \rangle$$

Problem Execution

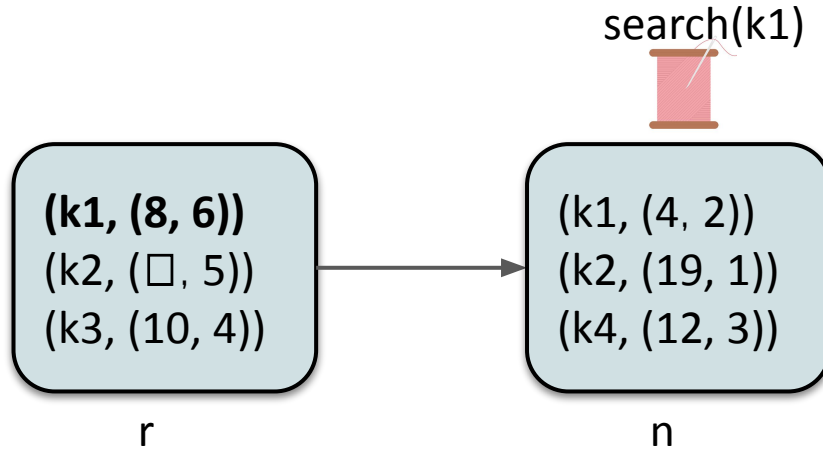


$$\langle M. \overline{MCS}(r, M) \rangle \text{ search } r \ k \langle v. \overline{MCS}(r, M) * M(k) = v \rangle$$

Problem Execution

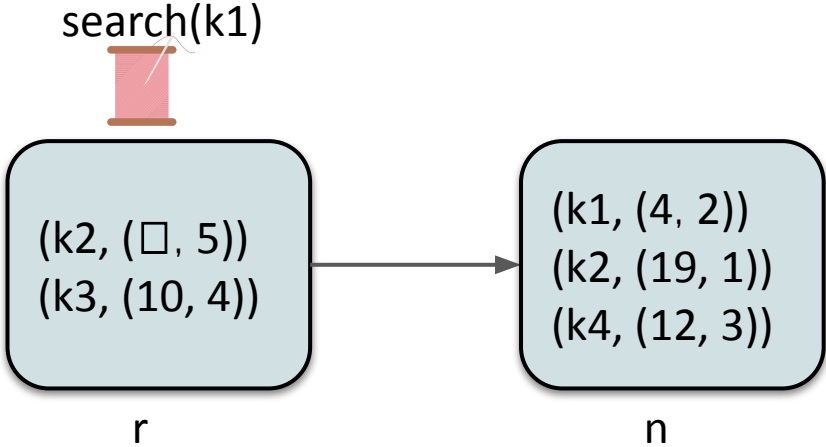
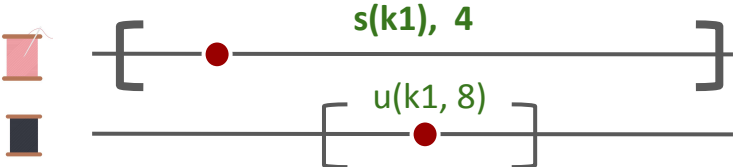


search(k1) finds 4, but $M(k1) = 8$



$$\langle M. \overline{\text{MCS}}(r, M) \rangle \text{ search } r \text{ } k \langle v. \overline{\text{MCS}}(r, M) * M(k) = v \rangle$$

Problem Execution



$$\langle M. \overline{MCS}(r, M) \rangle \text{ search } r \text{ } k \langle v. \overline{MCS}(r, M) * M(k) = v \rangle$$

Insight: Search Recency

Let (v_0, t_0) = most recent copy of k when search begins.

Then, search either returns:

1) (v_0, t_0) or

2) some (v, t) such that $t > t_0$.

Insight: Search Recency

Let (v_0, t_0) = most recent copy of k when search begins.

Then, search either returns:

1) (v_0, t_0) or

2) some (v, t) such that $t > t_0$.



Linearize at the beginning

Insight: Search Recency

Let (v_0, t_0) = most recent copy of k when search begins.

Then, search either returns:

1) (v_0, t_0) or

2) some (v, t) such that $t > t_0$.



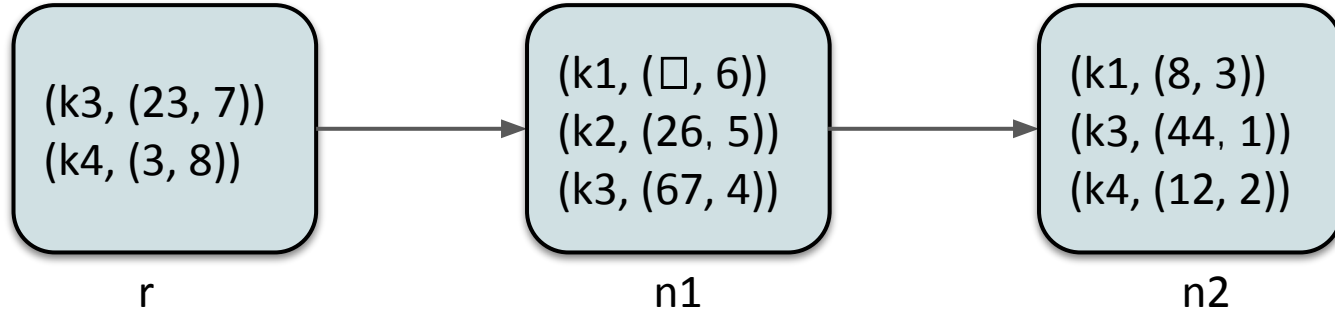
Linearize at the beginning



Require helping

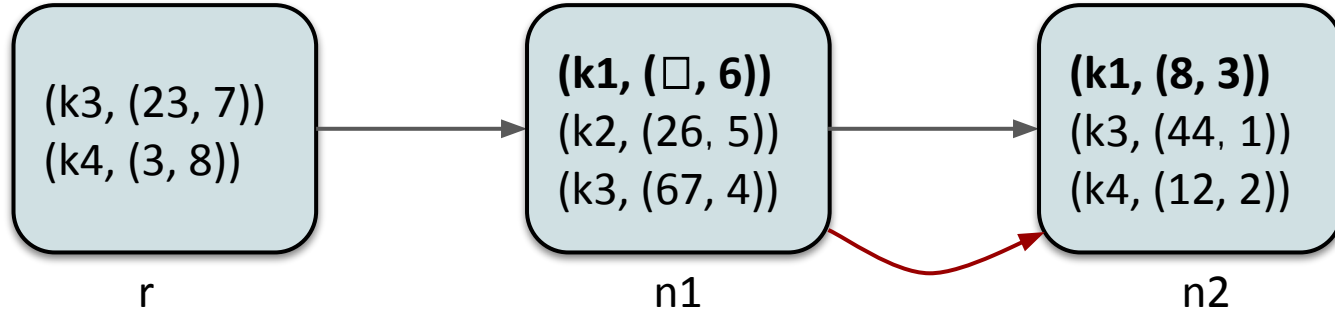
Invariant

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



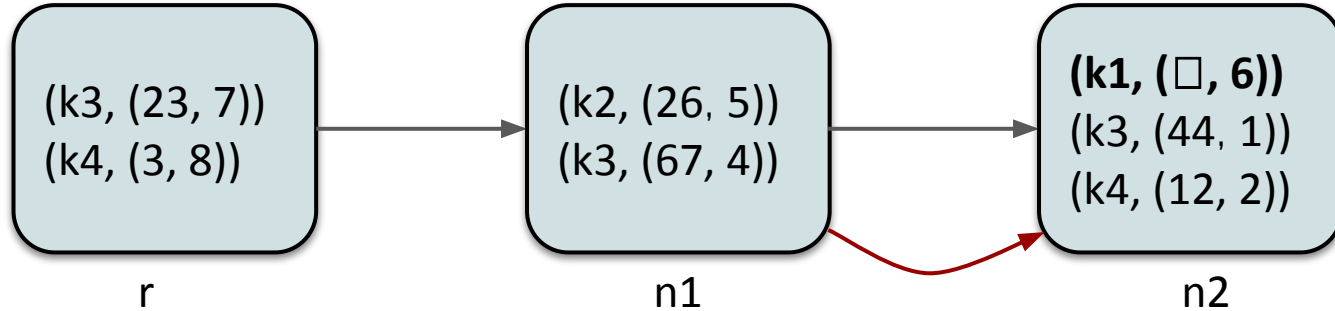
Invariant

“first copy reachable from the root is the most recent”




Invariant

“first copy reachable from the root is the most recent”



LSM DAG Template

```
let search r k = traverse r k
```

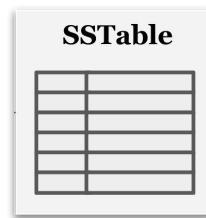
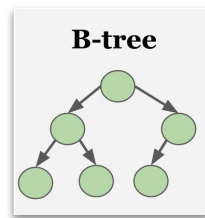
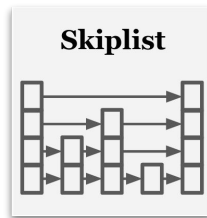
```
let rec traverse n k =  
  lockNode n;  
  match inContents n k with  
  | Some v -> unlockNode n; v  
  | None ->  
    match findNext n k with  
    | Some n' ->  
      unlockNode n;  
      traverse n' k  
    | None -> unlockNode n; 
```

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

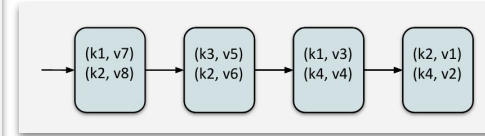
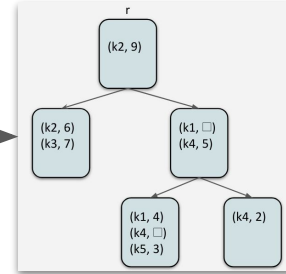
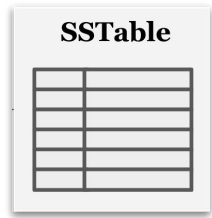
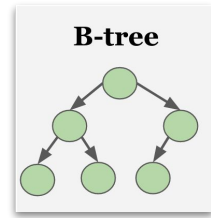
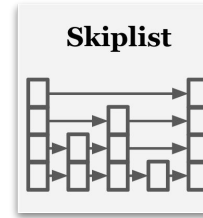


LSM DAG Template

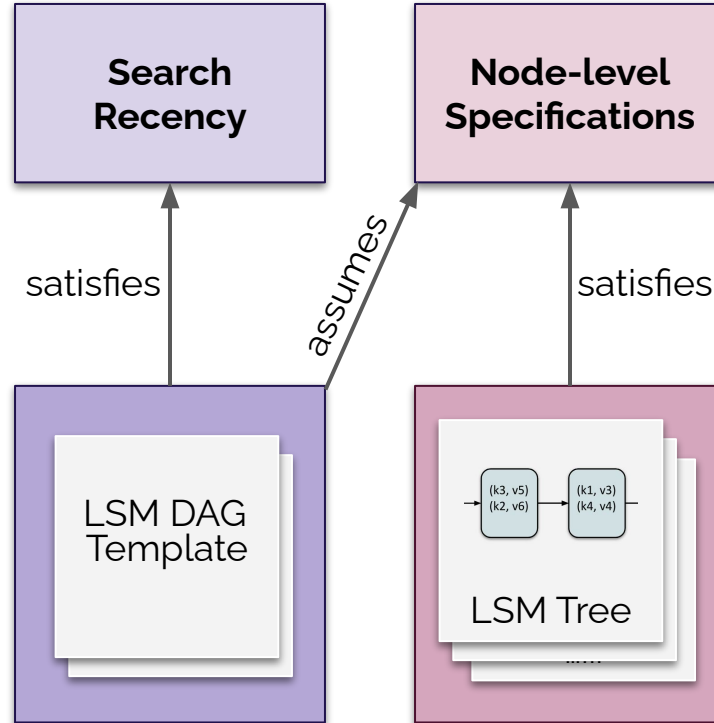
```
let search r k = traverse r k
```

```
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; 
```

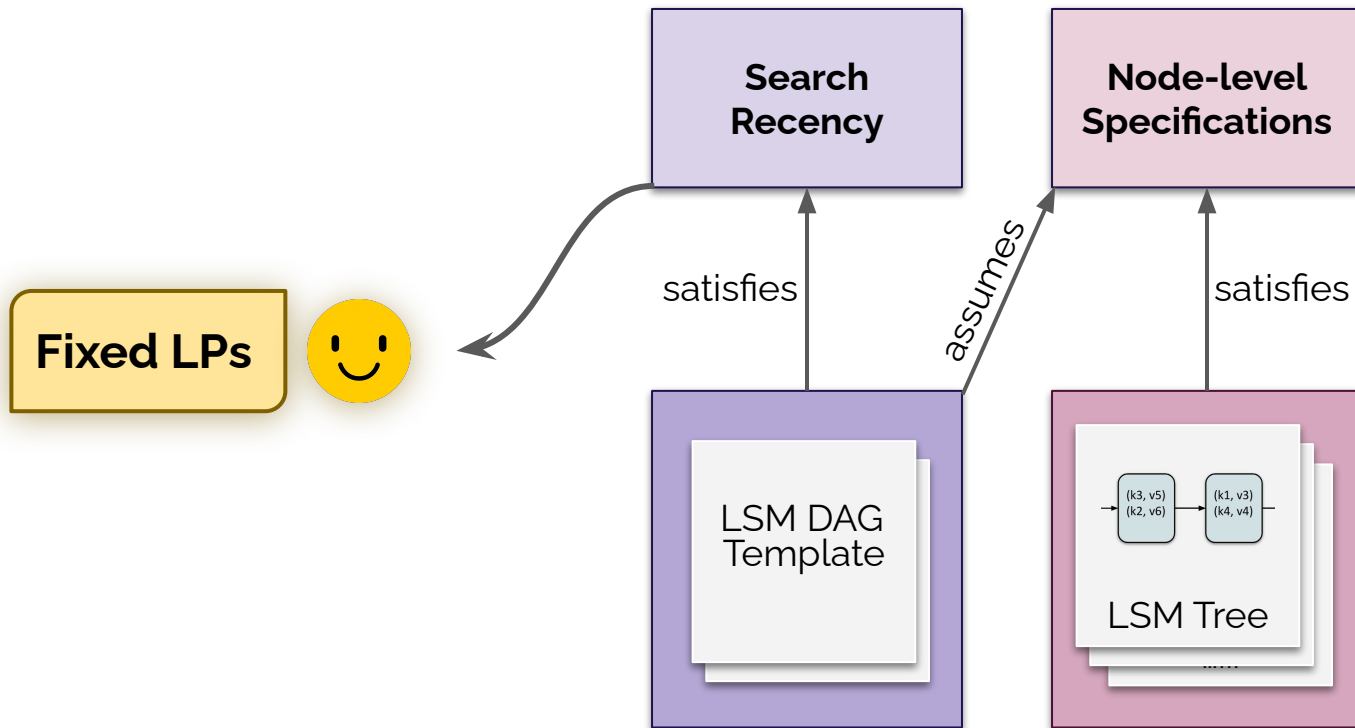
Helper function



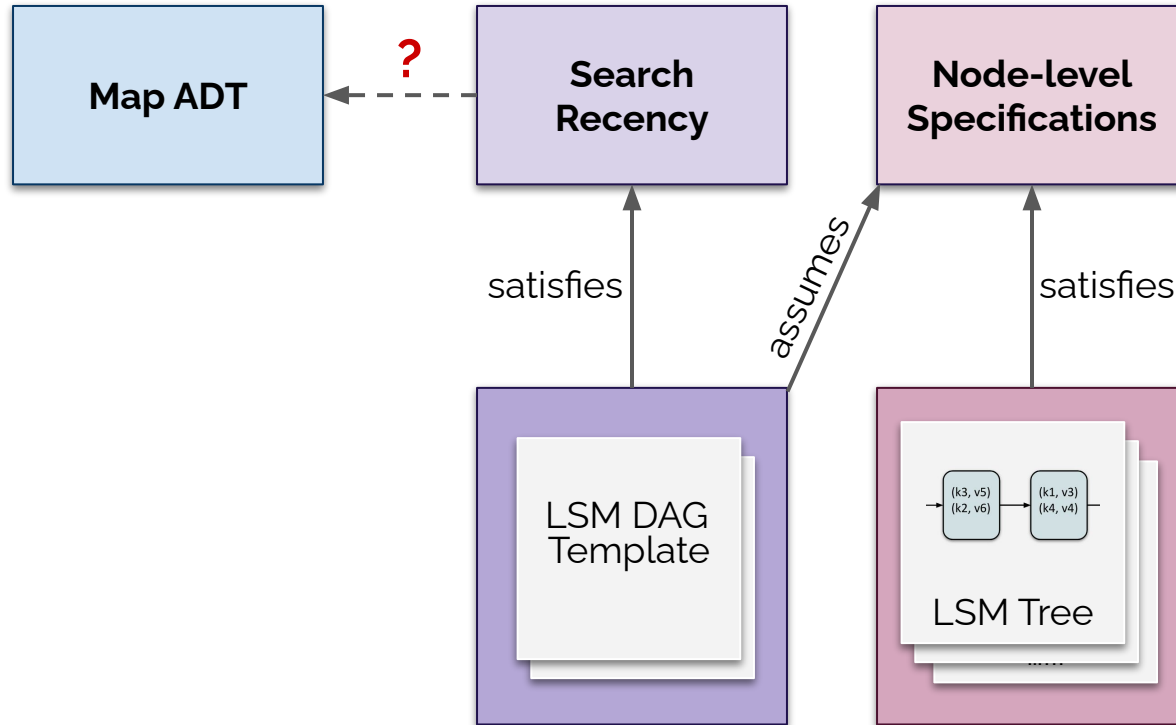
Overview



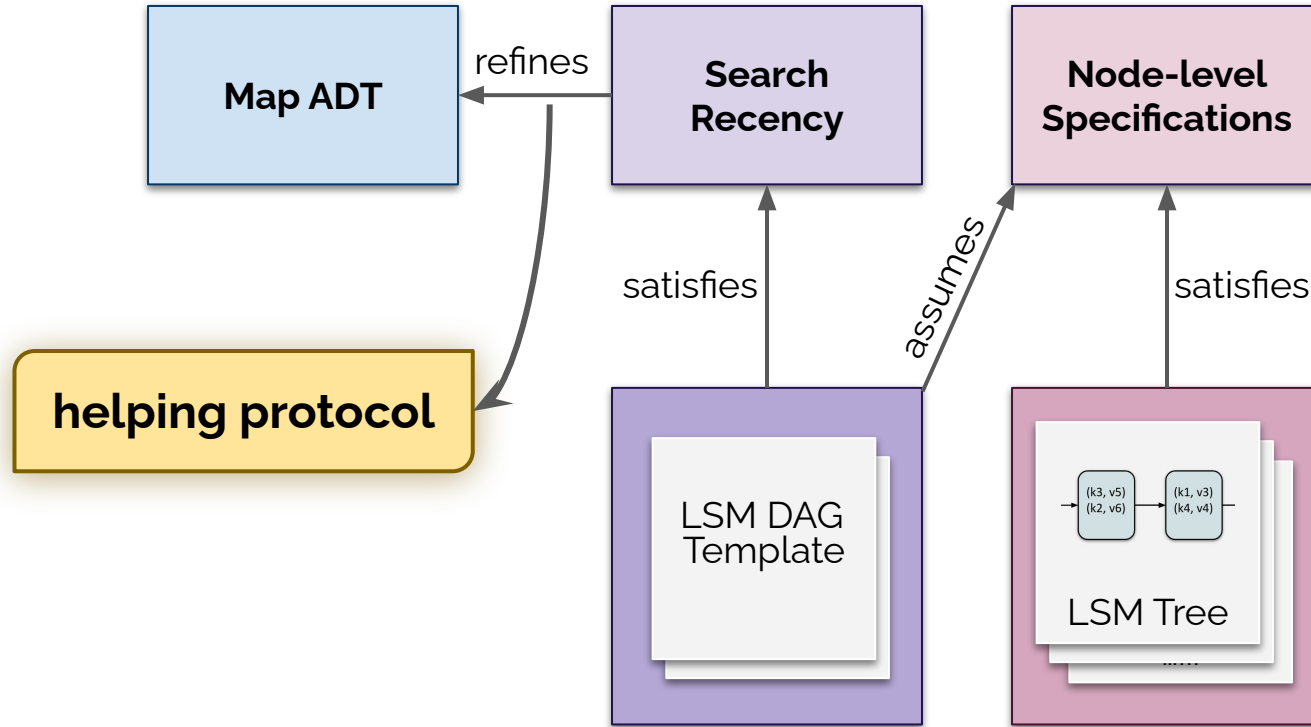
Overview



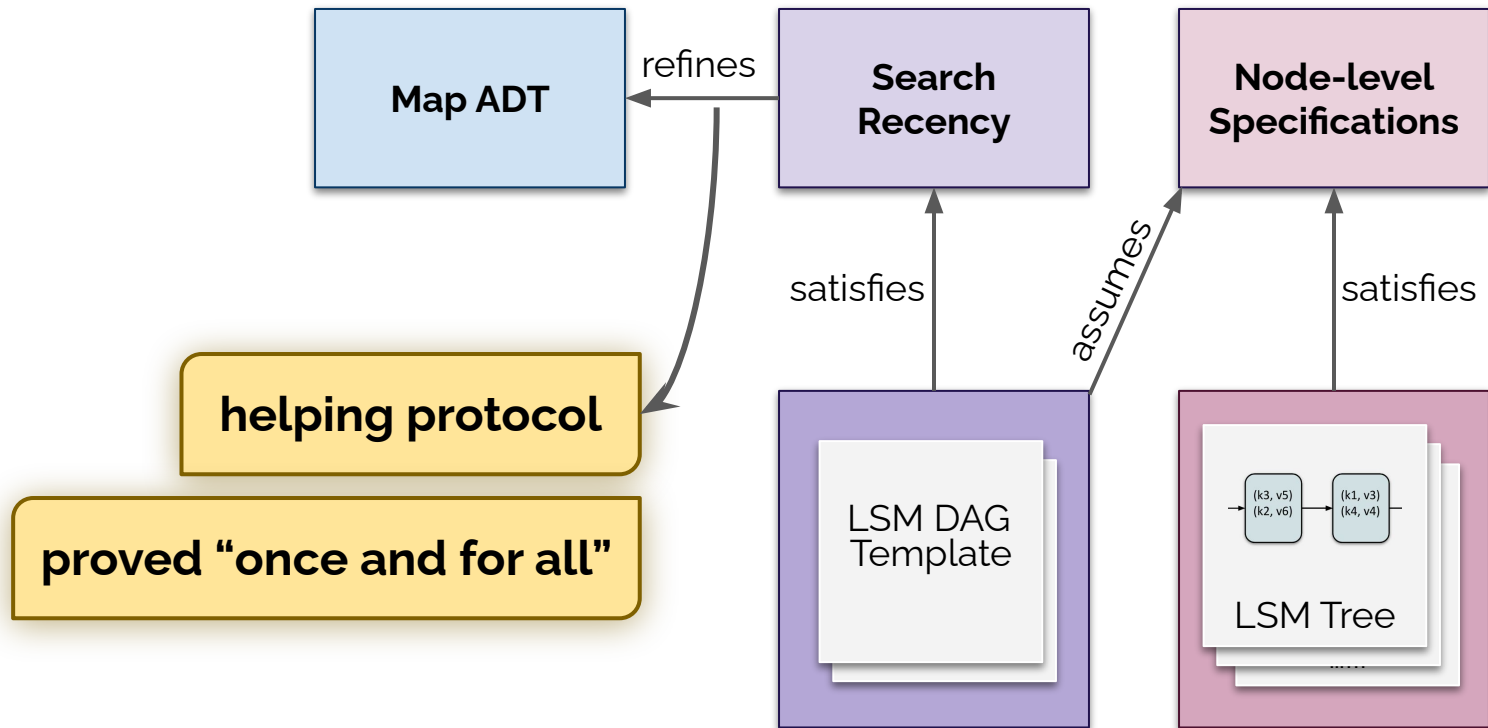
Overview



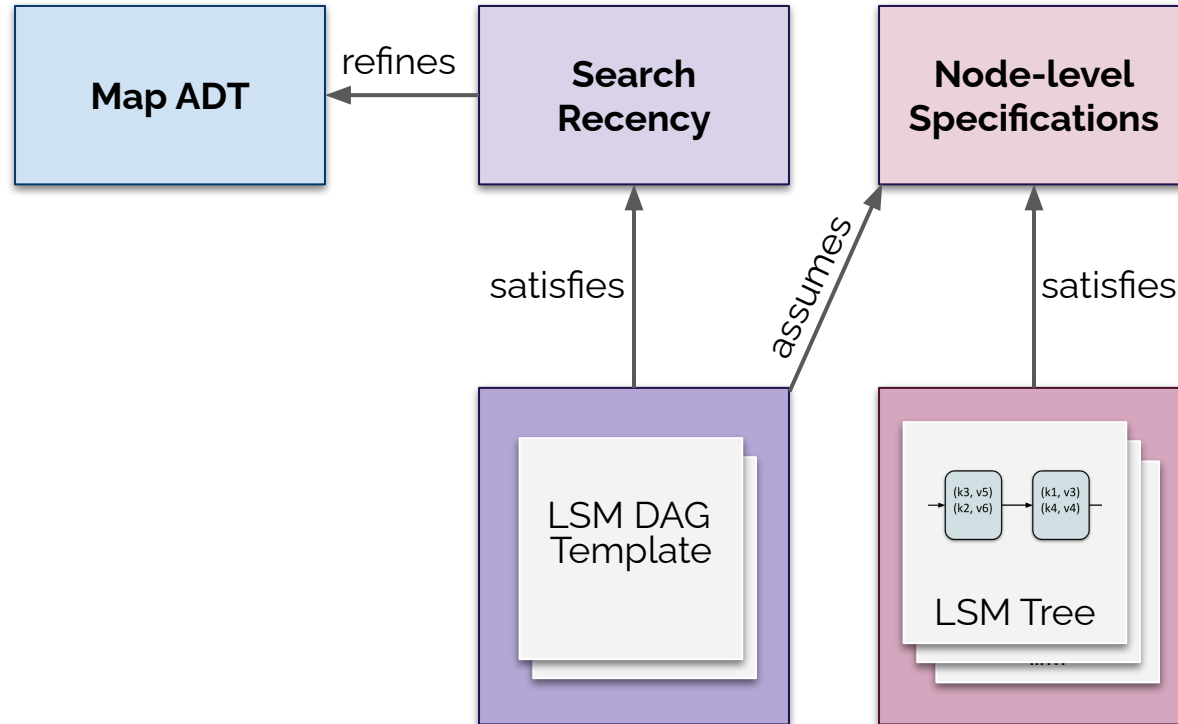
Overview



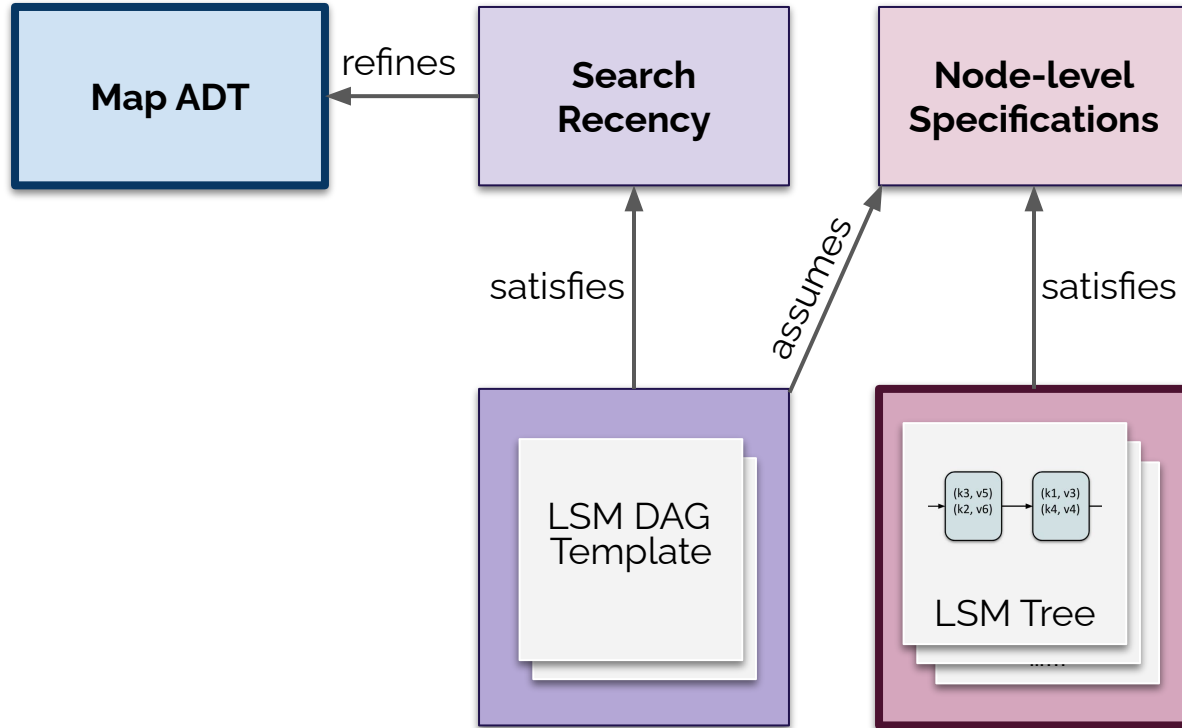
Overview



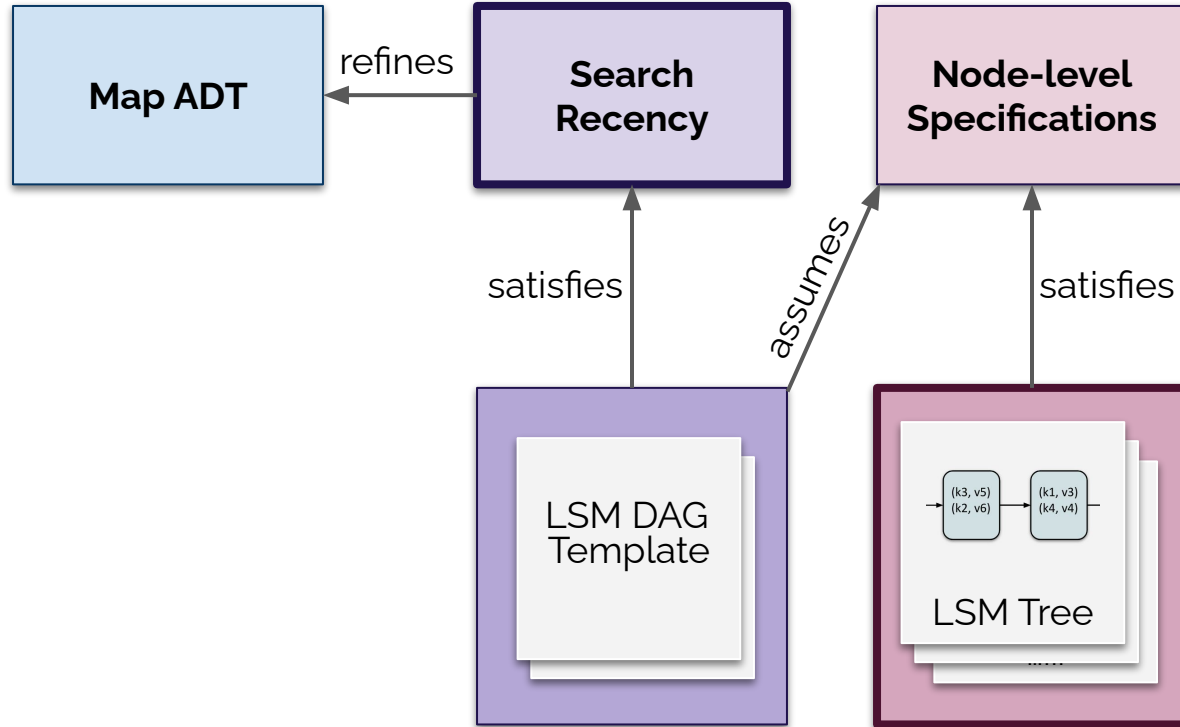
Summary



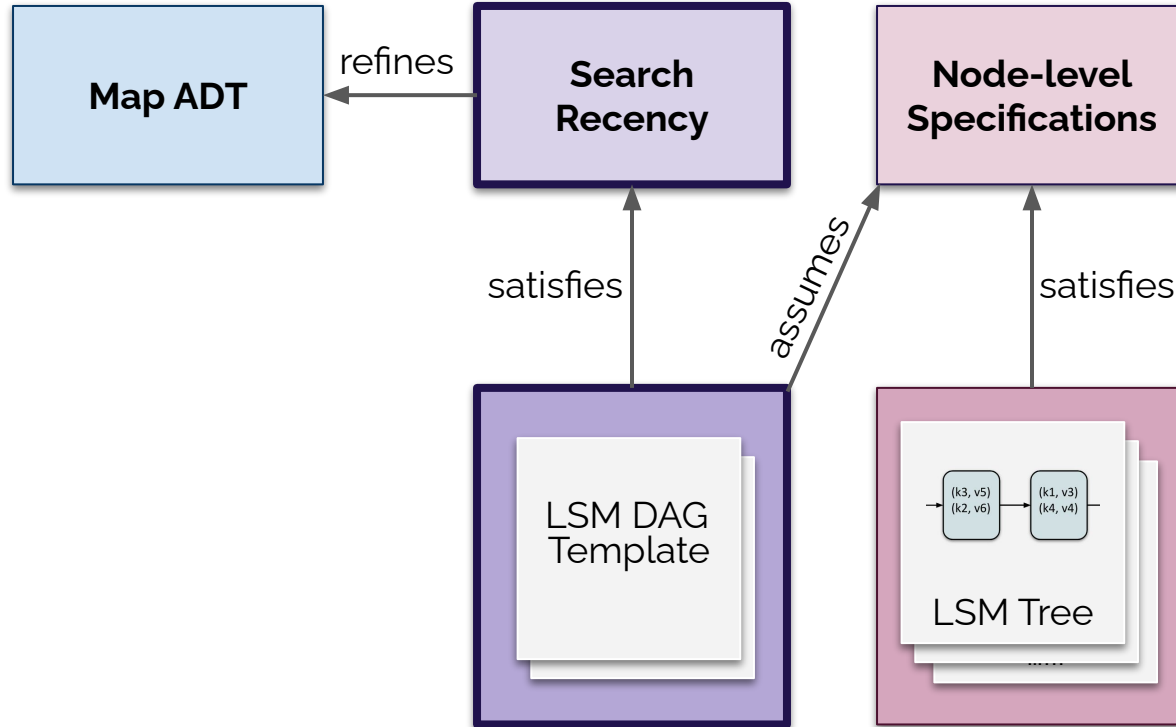
Summary



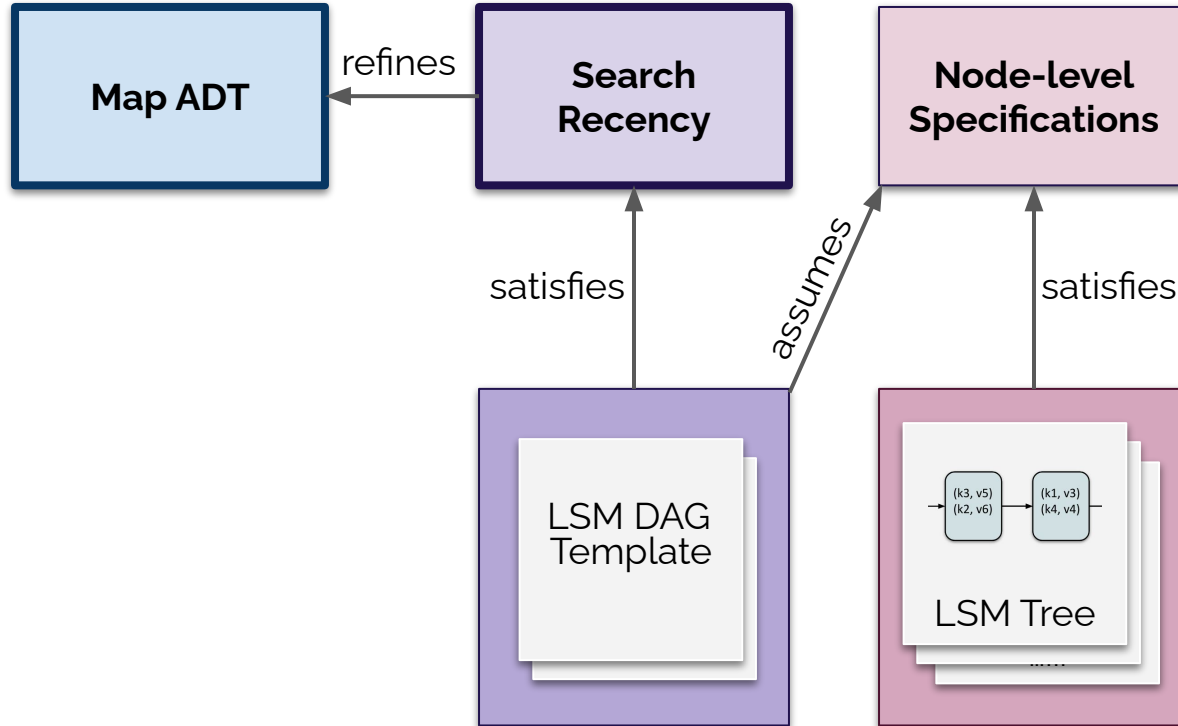
Summary



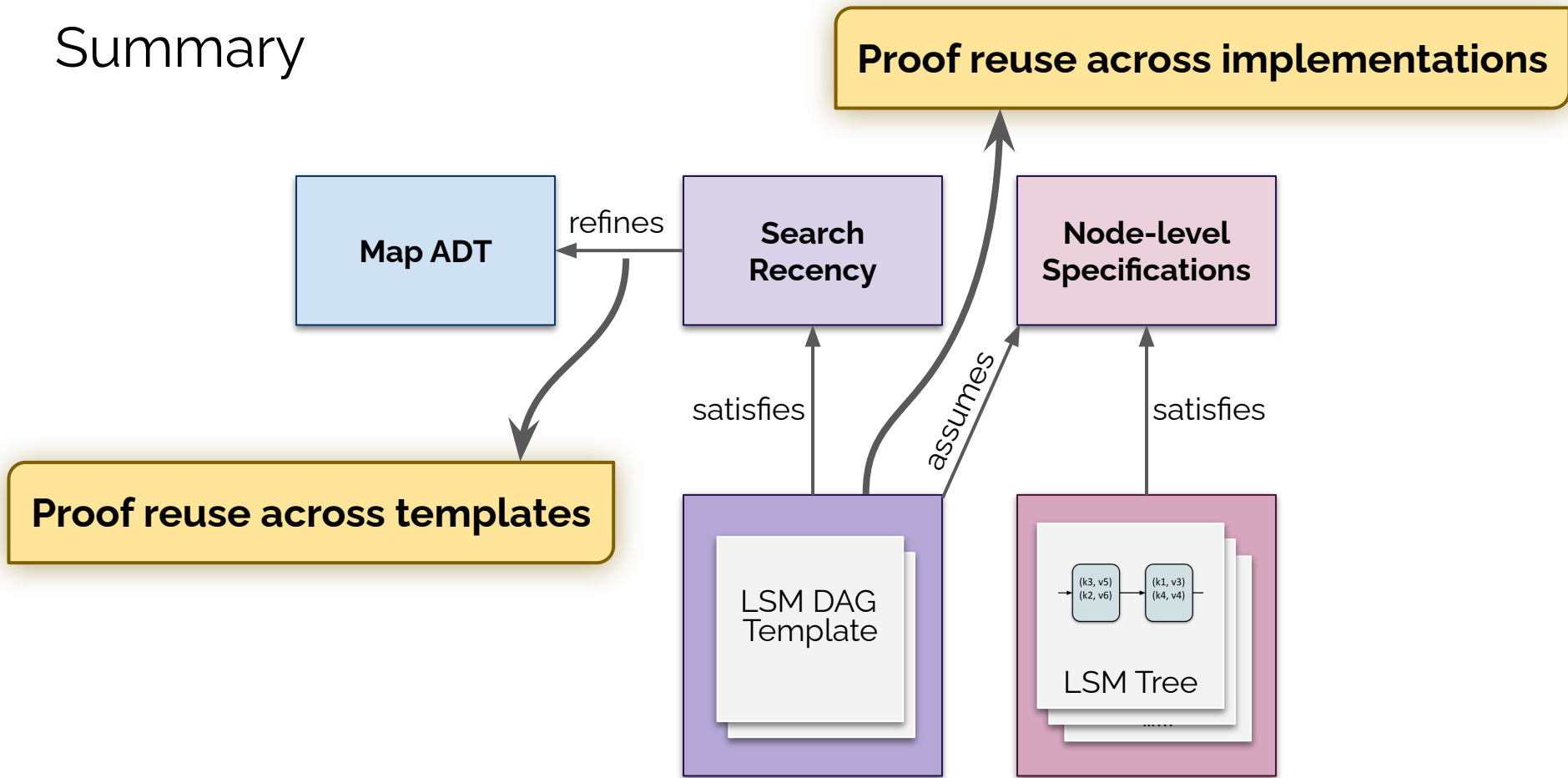
Summary



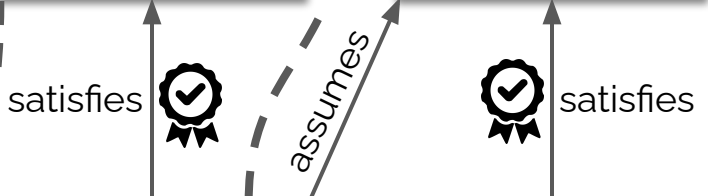
Summary



Summary

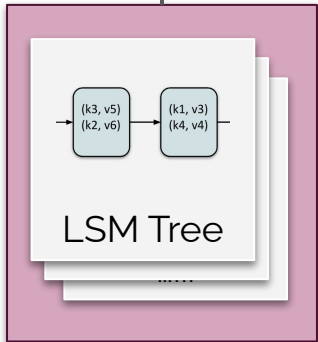


Proof reuse across implementations



Proof reuse across templates

Fixed LPs



Evaluation

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



Maintenance operation included!

Evaluation

Templates (Iris/Coq)

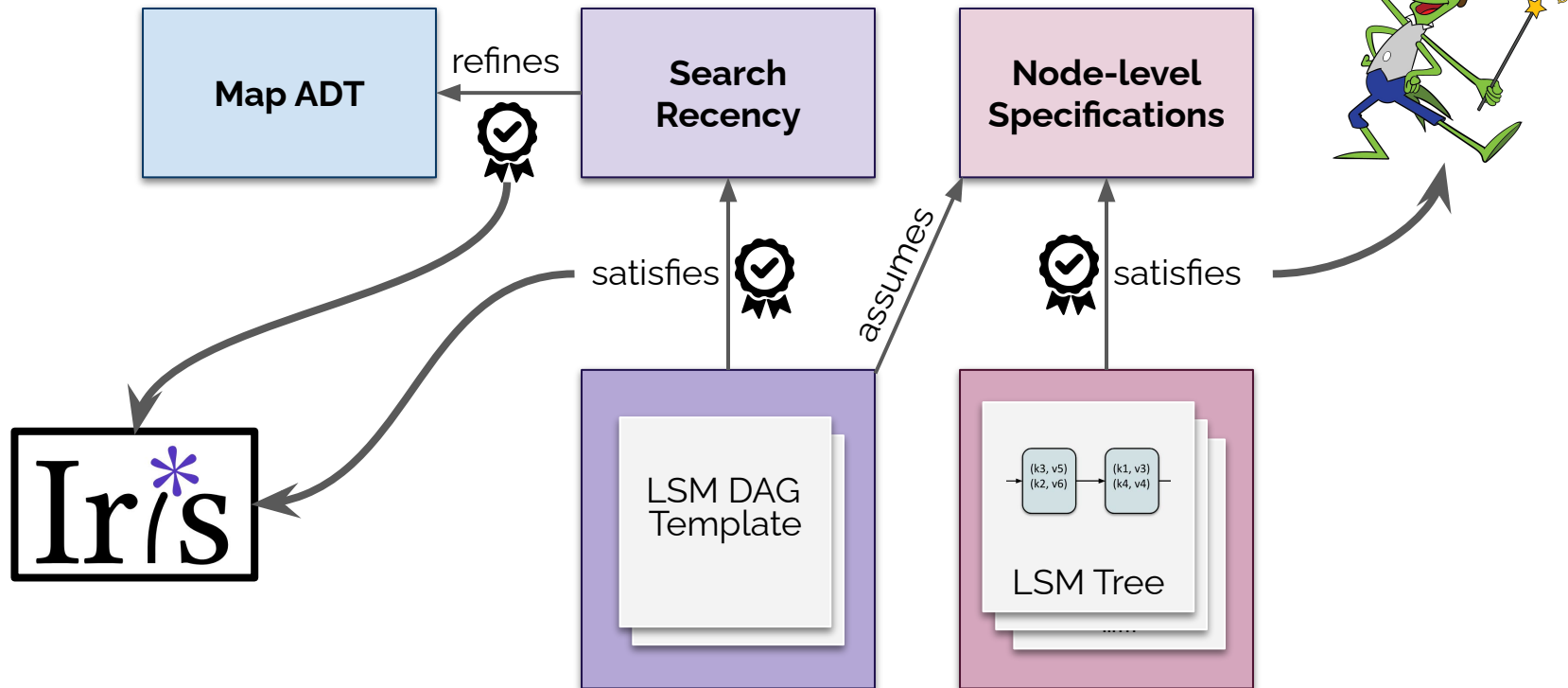
Module	Code	Proof	Total	Time
Flow Library	0	3757	3757	41
Lock Implementation	10	333	343	10
Client-level Spec	2	792	794	31
DF Template	26	934	960	68
LSM DAG Template	46	3587	3633	307
Total	84	9403	9487	457

Implementations (GRASShopper)

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

Summary

GRASShopper



Thank you!

