CoLoSL
Concurrent Local Subjective Logic

Azalea Raad Jules Villard Philippa Gardner

Imperial College London

24 April 2015

Global Shared Resources

$P_1 \wedge P_2$

$$\frac{\left\{\boxed{P1}\right\} C1 \left\{\boxed{Q1}\right\} \qquad \left\{\boxed{P2}\right\} C2 \left\{\boxed{Q2}\right\}}{\left\{\boxed{P1 \;\wedge\; P2}\right\} C1 \parallel C2 \left\{\boxed{Q1 \wedge Q2}\right\}}$$

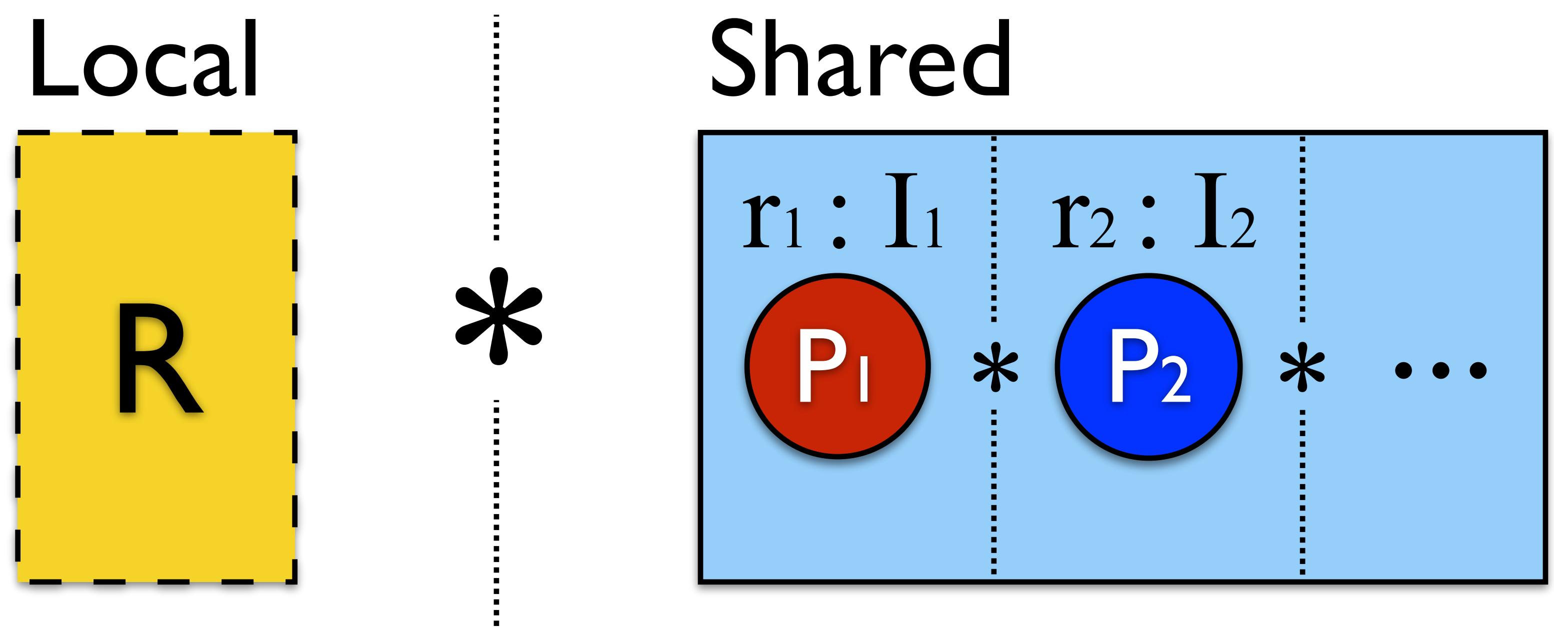
Owicki-Gries, Rely-Guarantee

Global Shared Resources

$$\frac{\left\{\boxed{P1}\right\}\ C1\ \left\{\boxed{Q1}\right\} \qquad \left\{\boxed{P2}\right\}\ C2\ \left\{\boxed{Q2}\right\}}{\left\{\boxed{P1\ \wedge\ P2}\right\}\ C1 \parallel C2\ \left\{\boxed{Q1\ \wedge\ Q2}\right\}}$$

- **No framing on shared resources / interference**
  - Reasoning on GLOBAL resources
  - Interference on ALL resources considered
- **No extension**
  - cannot dynamically share resources/extend interference

Disjoint Shared Resources



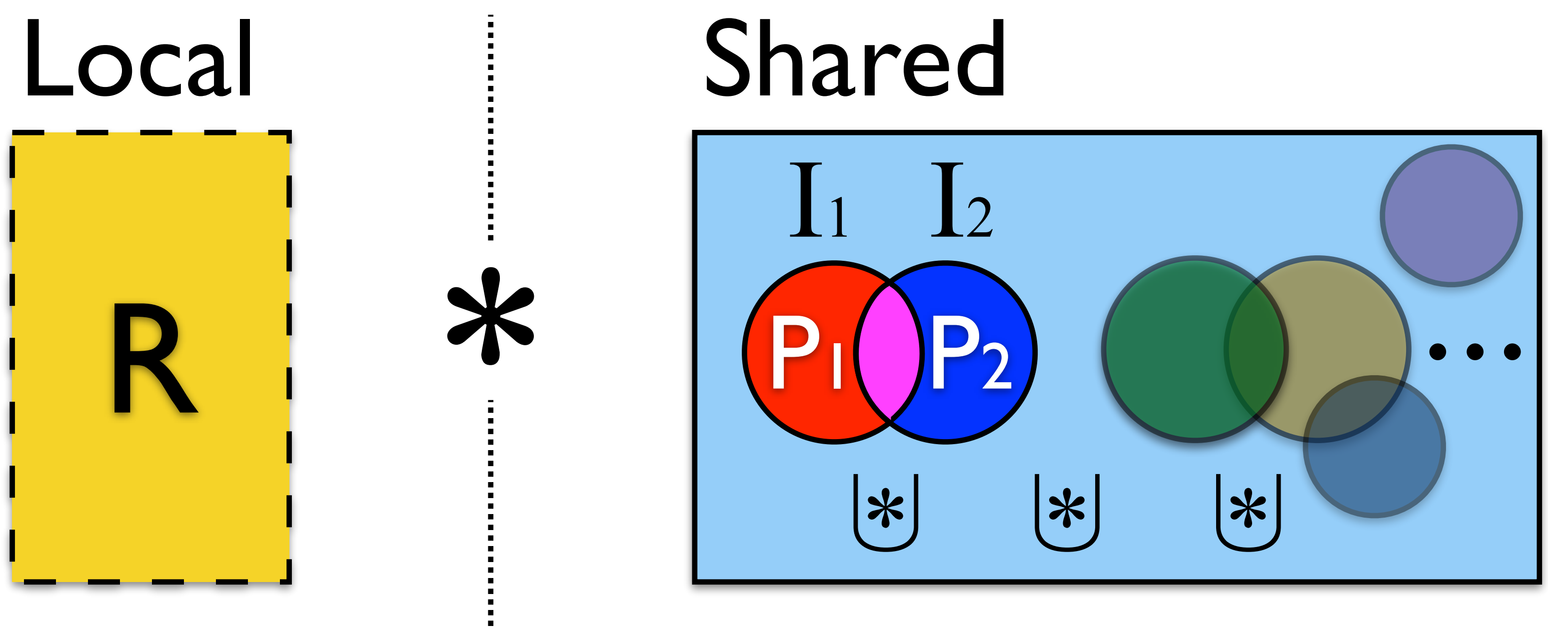
CSL, FCSL, CAP, HOCAP, iCAP, TaDA

Disjoint Shared Resources

$$\frac{\left\{\boxed{P1}_{I_1}\right\}\ C1\ \left\{\boxed{Q1}_{I_1}\right\} \qquad \left\{\boxed{P2}_{I_2}\right\}\ C1\ \left\{\boxed{Q2}_{I_2}\right\}}{\left\{\boxed{P1}_{I_1} * \boxed{P2}_{I_2}\right\}\ C1 \parallel C2\ \left\{\boxed{Q1}_{I_1} * \boxed{Q2}_{I_2}\right\}}$$

- **<u>Limited framing on shared resources / interference</u>**
  - **Static** (pre-determined) frames (regions/ invariants)
  - **Physically** disjoint frames
- **<u>Limited extension</u>**
  - Can create new regions / invariants
  - **Cannot** extend regions with more resources/invariants

CoLoSL: Concurrent Local Subjective Logic



CoLoSL

CoLoSL: <u>**Co**</u>ncurrent <u>**Lo**</u>cal <u>**S**</u>ubjective <u>**L**</u>ogic

$$\frac{\left\{ \boxed{\mathrm{P1}}_{\mathrm{I1}} \right\} \ \mathrm{C1} \ \left\{ \boxed{\mathrm{Q1}}_{\mathrm{I1}} \right\} \qquad \left\{ \boxed{\mathrm{P2}}_{\mathrm{I2}} \right\} \ \mathrm{C1} \ \left\{ \boxed{\mathrm{Q2}}_{\mathrm{I2}} \right\}}{\left\{ \boxed{\mathrm{P1}}_{\mathrm{I1}} \uplus\!\!\!\!\ast \ \boxed{\mathrm{P2}}_{\mathrm{I2}} \right\} \ \mathrm{C1} \parallel \mathrm{C2} \ \left\{ \boxed{\mathrm{Q1}}_{\mathrm{I1}} \uplus\!\!\!\!\ast \ \boxed{\mathrm{Q2}}_{\mathrm{I2}} \right\}}$$

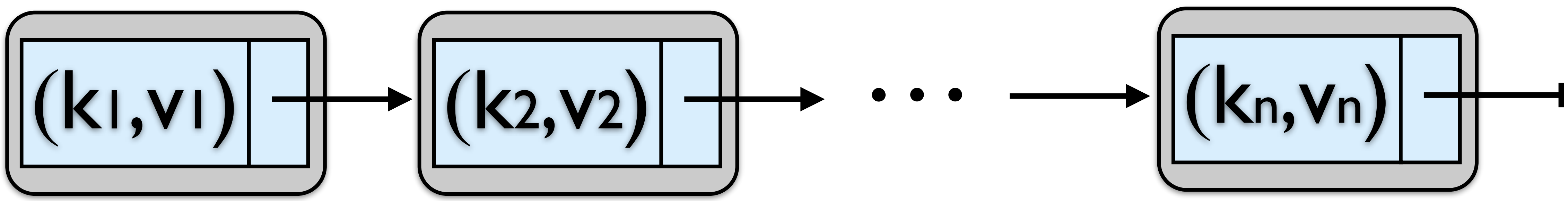
- <u>**Flexible framing on shared resources/invariants**</u>
  - **Overlapping** frames
  - **Dynamic** framing/rewriting of interference
- <u>**Dynamic extension**</u>
  - Extension of shared state with new resources/interference

Examples

- **B+ Tree** (thanks to Shale Xiong)
  - Module composition; proof modularity; better abstraction
  - B+ Tree = List + BSTree

Ordered Singly Linked-List

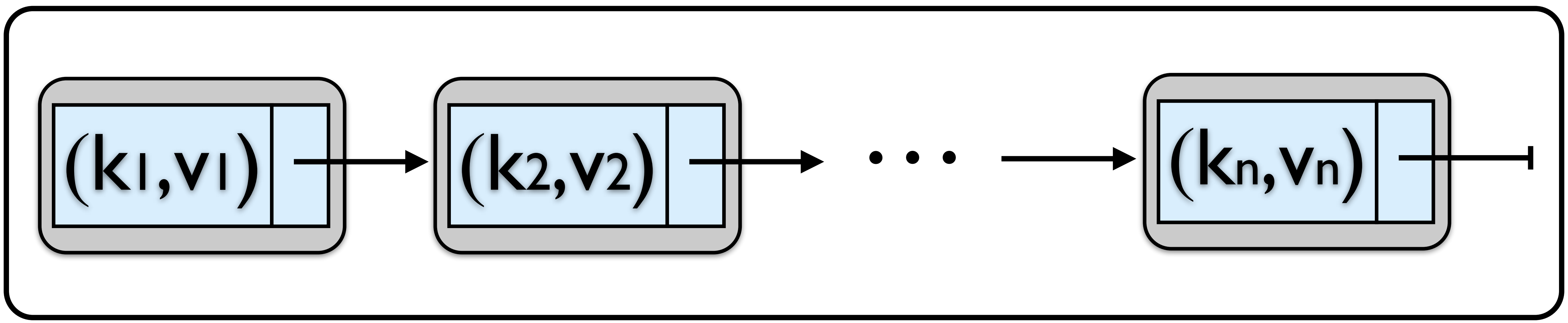
$\text{List}([(k_1, v_1), (k_2, v_2), \ldots, (k_n, v_n)]) =$



- **Value update**
- **Map** (e.g. increment all elements)
- **Insertion** (involves pointer surgery)
- **Removal** (involves pointer surgery)

Concurrent Ordered Singly Linked-List

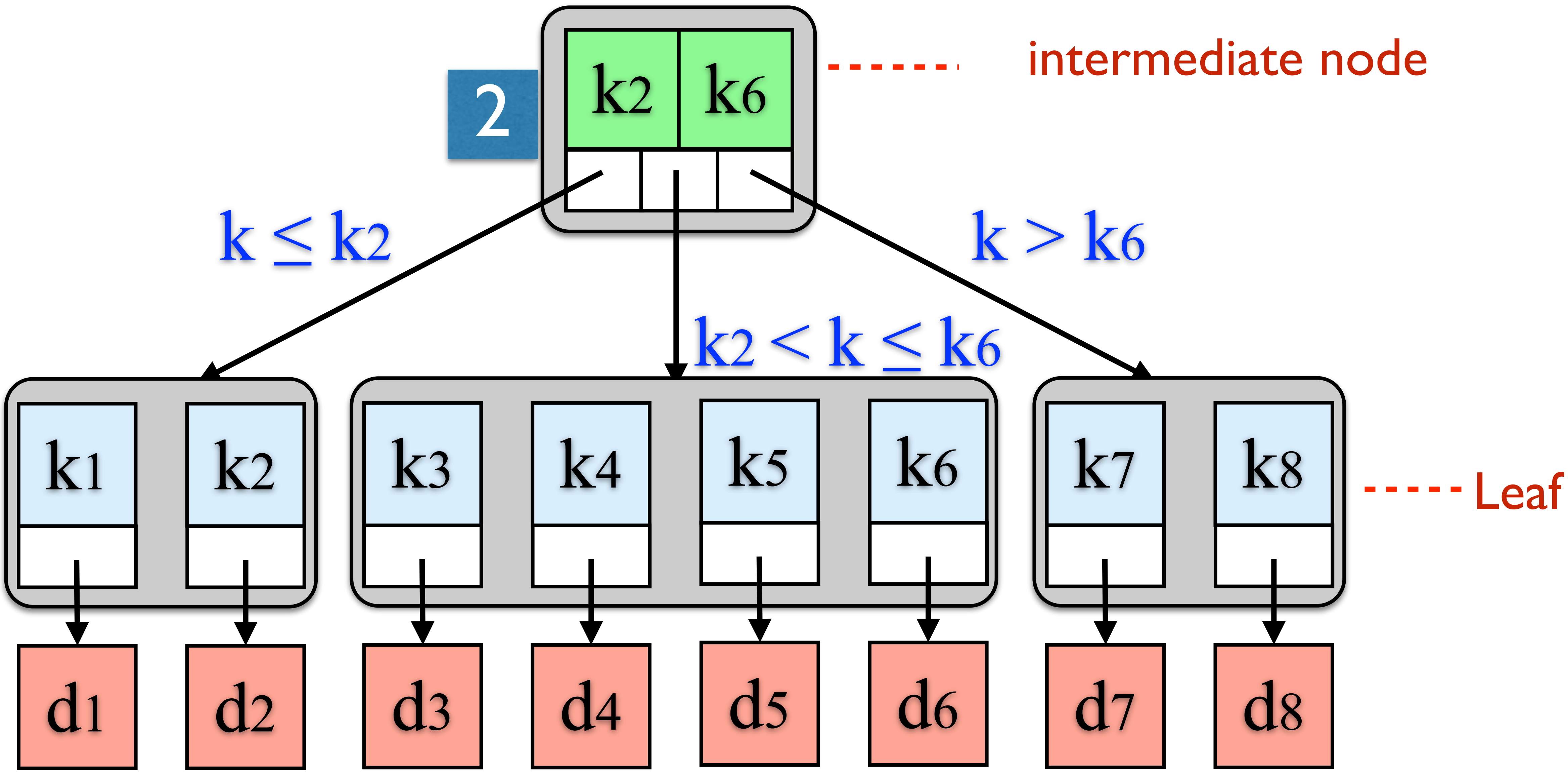
List([(k_1, v_1), (k_2, v_2), … , (k_n, v_n)]) =



$I_L = I_{up} \cup I_{map} \cup I_{add} \cup I_{rem}$

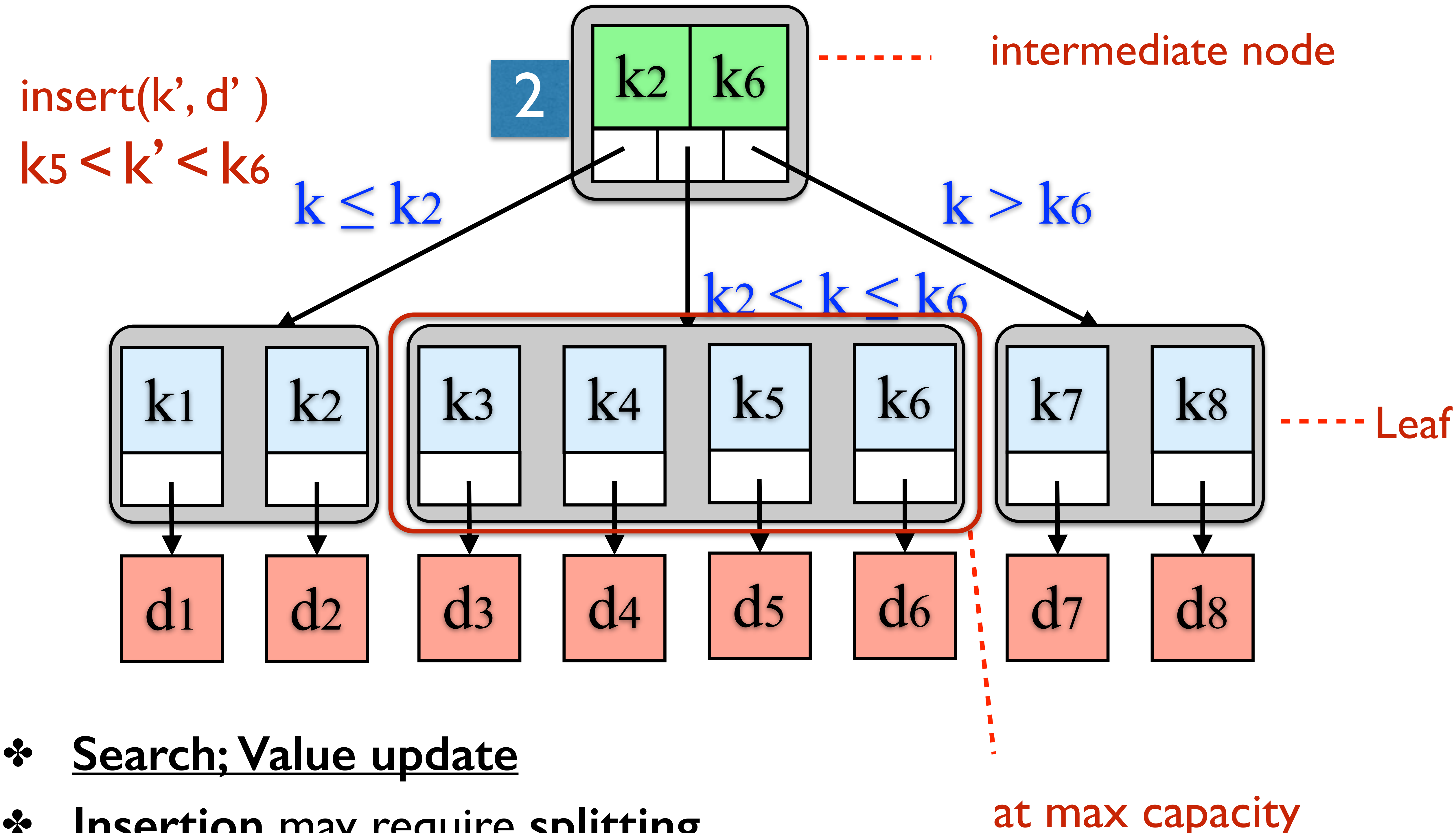
- **Value update**
- **Map** (e.g. increment all elements)
- **Insertion** (involves pointer surgery)
- **Removal** (involves pointer surgery)

Balanced Search Tree (Degree 2)



- **Balanced**: all immediate subtrees of a node have the same height
- **Leaf-heavy**: data (values) stored in leaf nodes
- **Degree (d)**: no. of children (m) on each node $d \leq m \leq 2d$

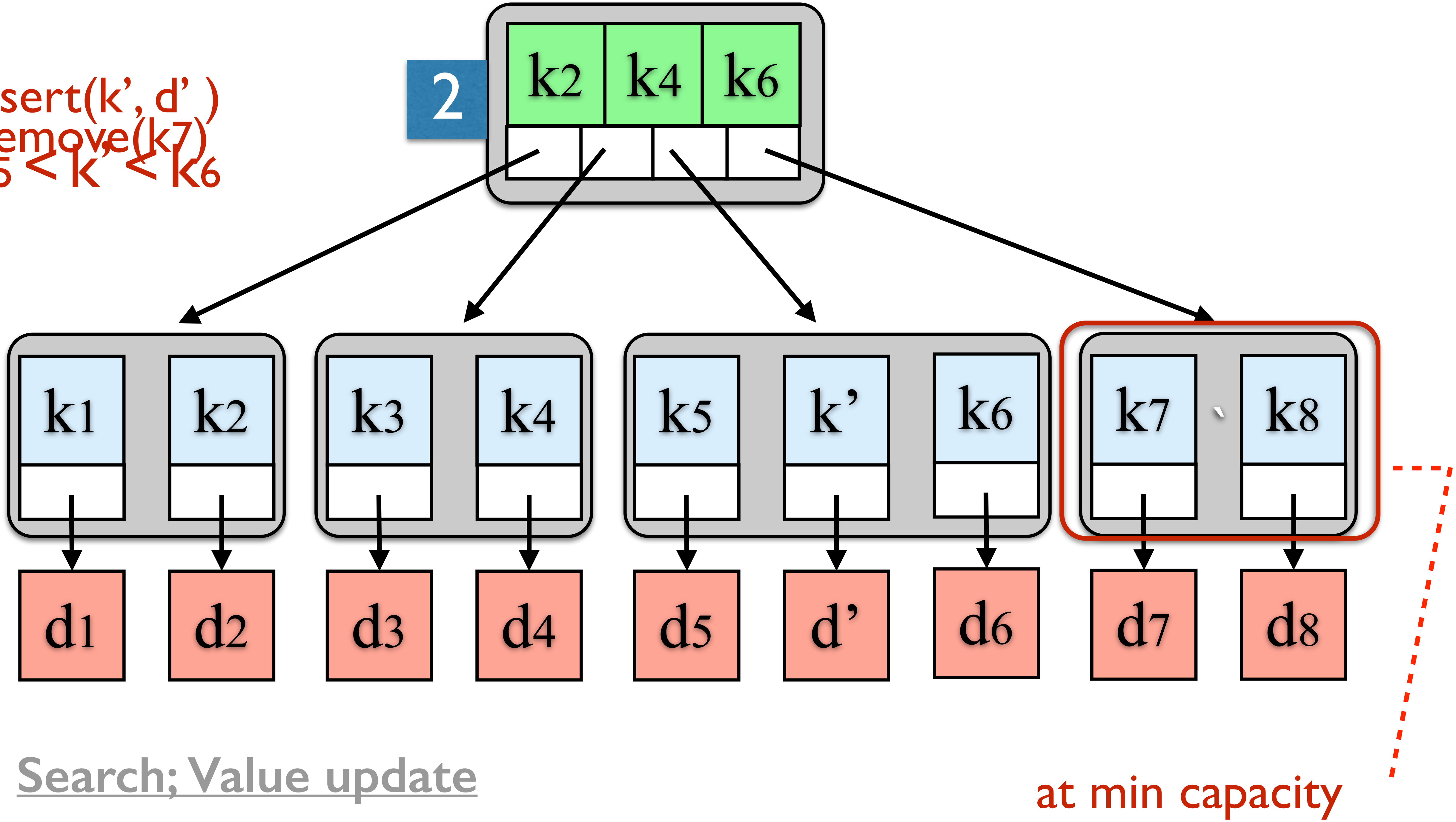
Balanced Search Tree (Degree 2)



- **Search; Value update**
- **Insertion** may require **splitting**
- **Removal**

Balanced Search Tree

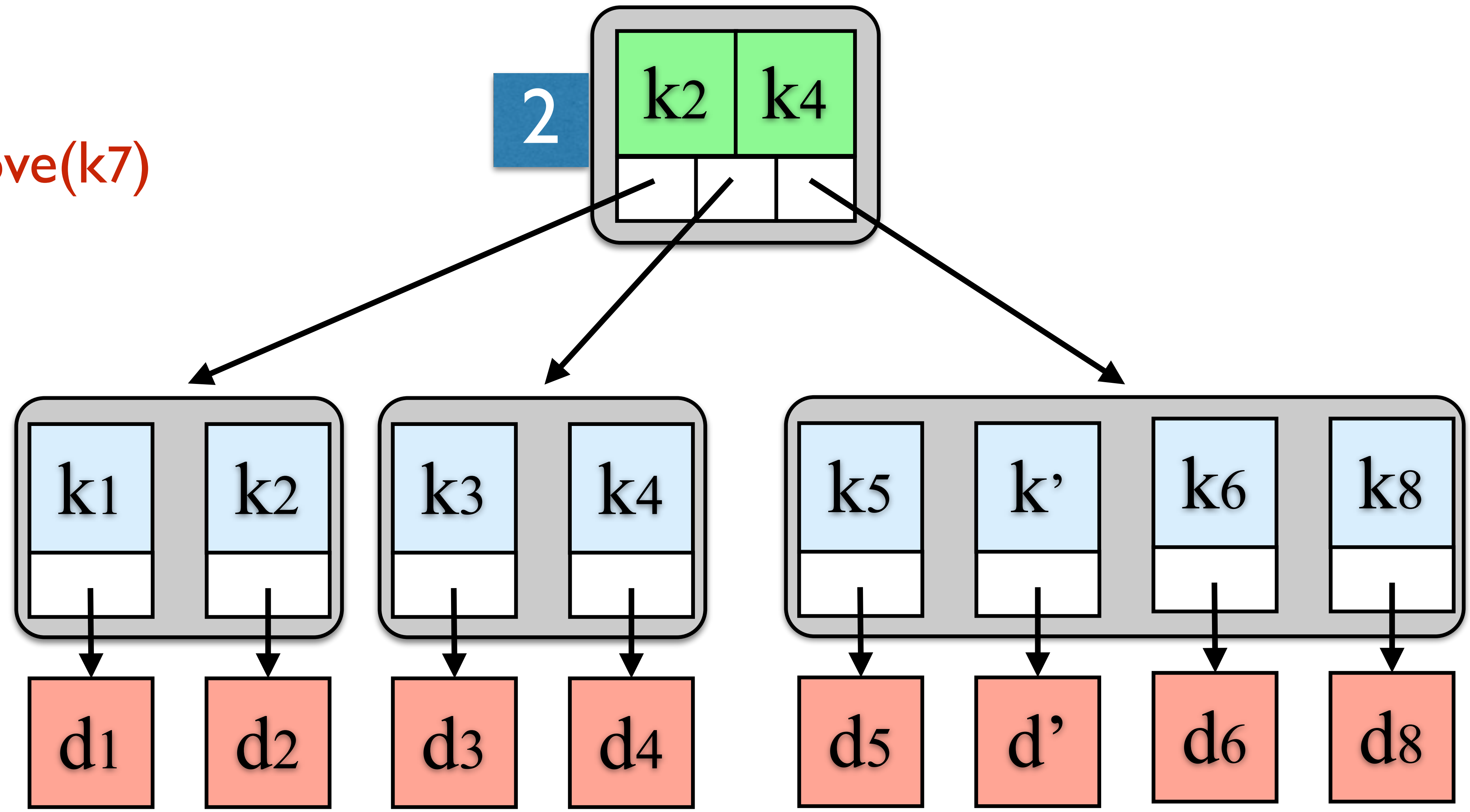
insert(k’, d’)
remove(k7)
k5 < k’ < k6



- Search; Value update
- **Insertion** may require **splitting**
- **Removal** may require **merging**

Balanced Search Tree

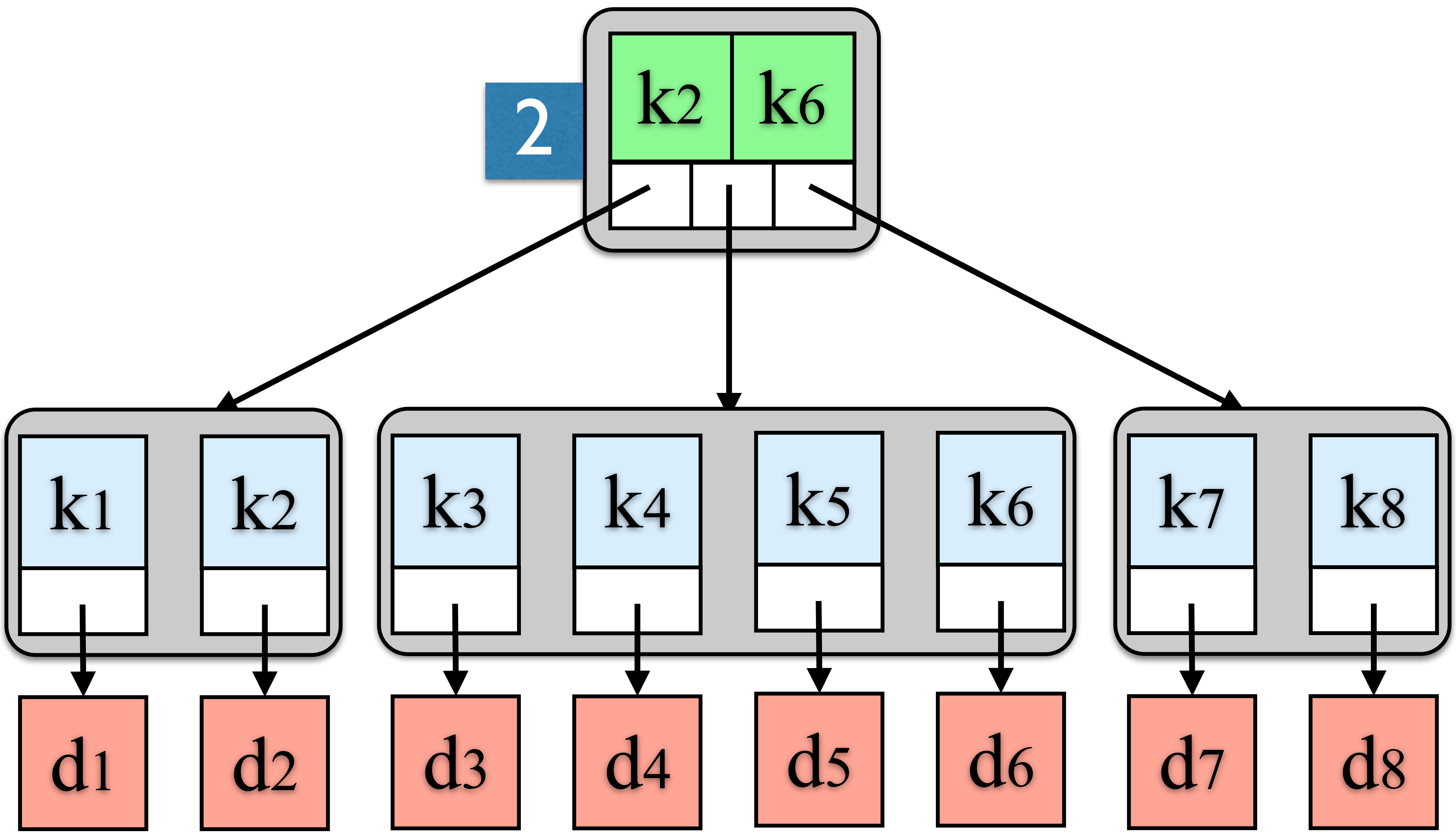
remove(k7)



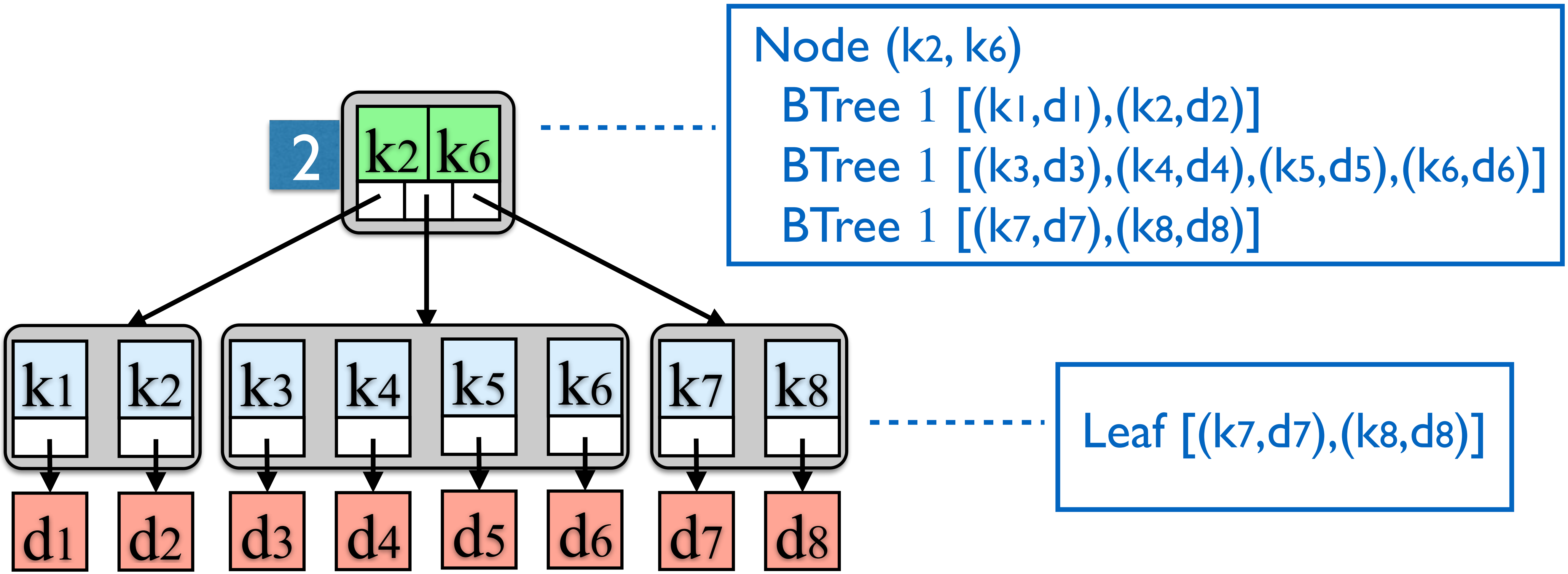
- Search; Value update
- Insertion
- **Removal** may require **merging**

Balanced Search Tree

BSTree 2 ([(k1, d1) … (k8, d8)]) =



Balanced Search Tree



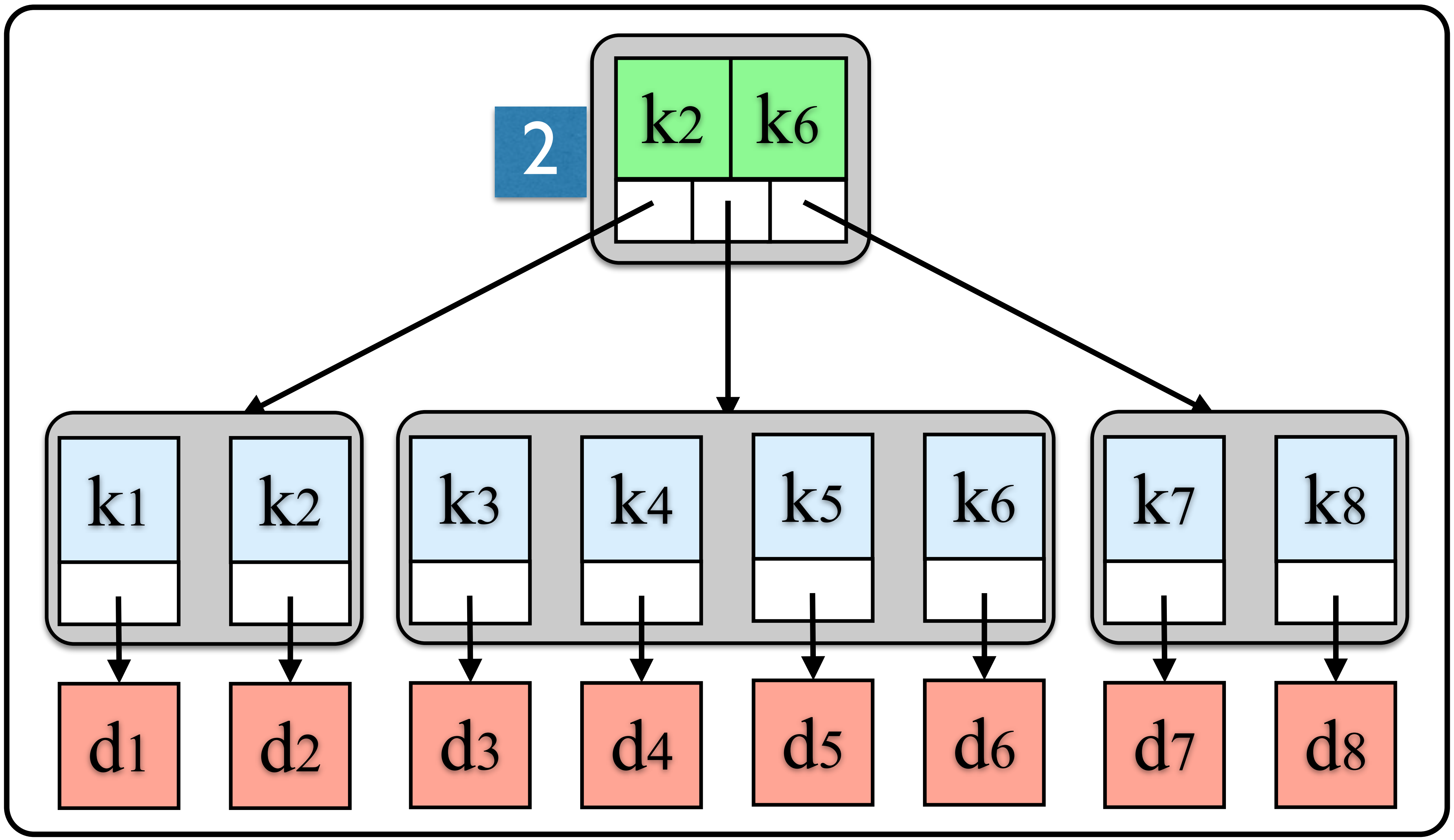
BSTree (h+1) L =

$\exists\, k_1, \ldots, k_m, L_1, \ldots, L_{(m+1)}$. $L = L_1 \uplus \ldots \uplus L_{(m+1)}$
Node ($k_1, \ldots, k_m$) (BSTree h L_1) … (BSTree h $L_{(m+1)}$)

BSTree 1 L = Leaf L

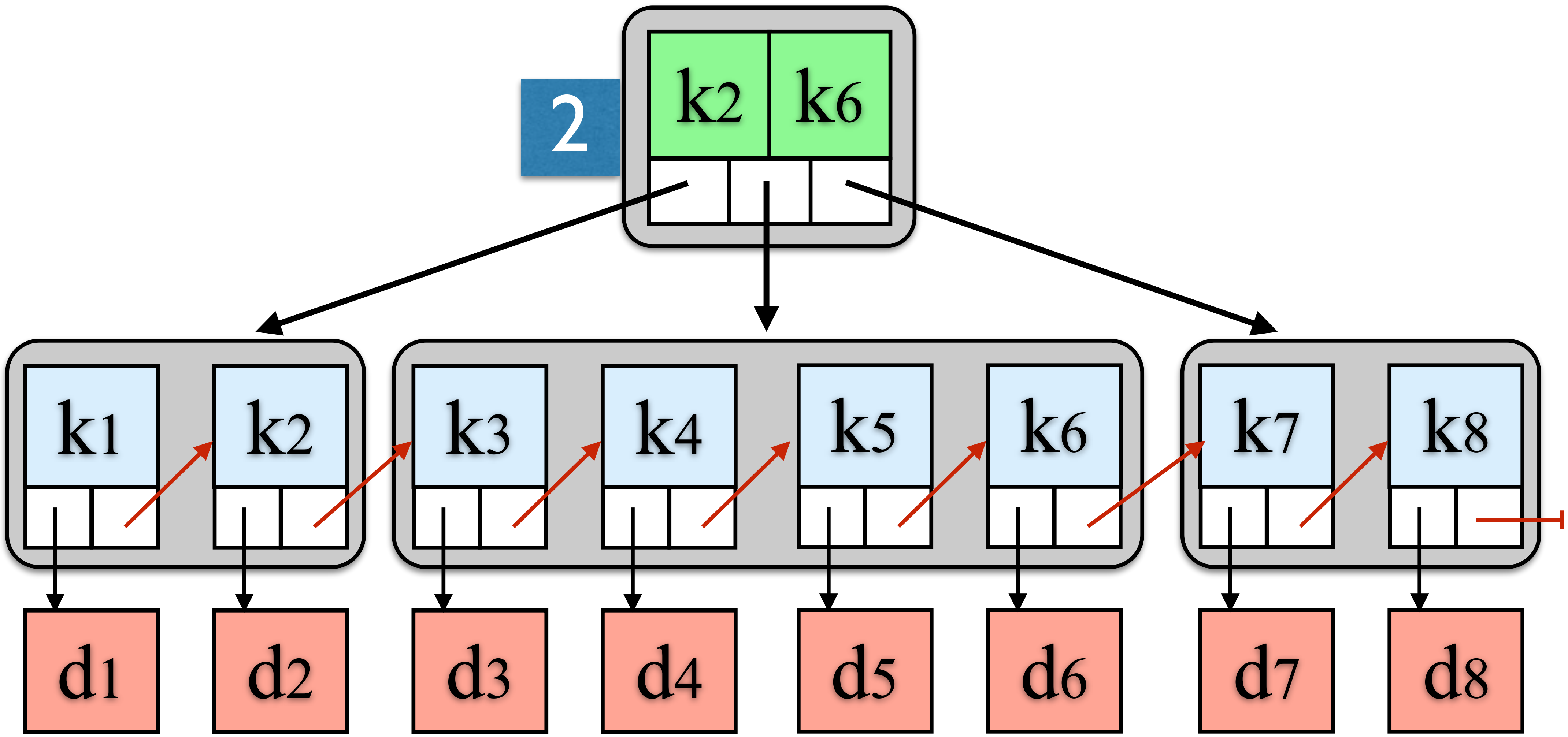
Concurrent Balanced Search Tree

BSTree([(k1, d1) … (k8, d8)]) =



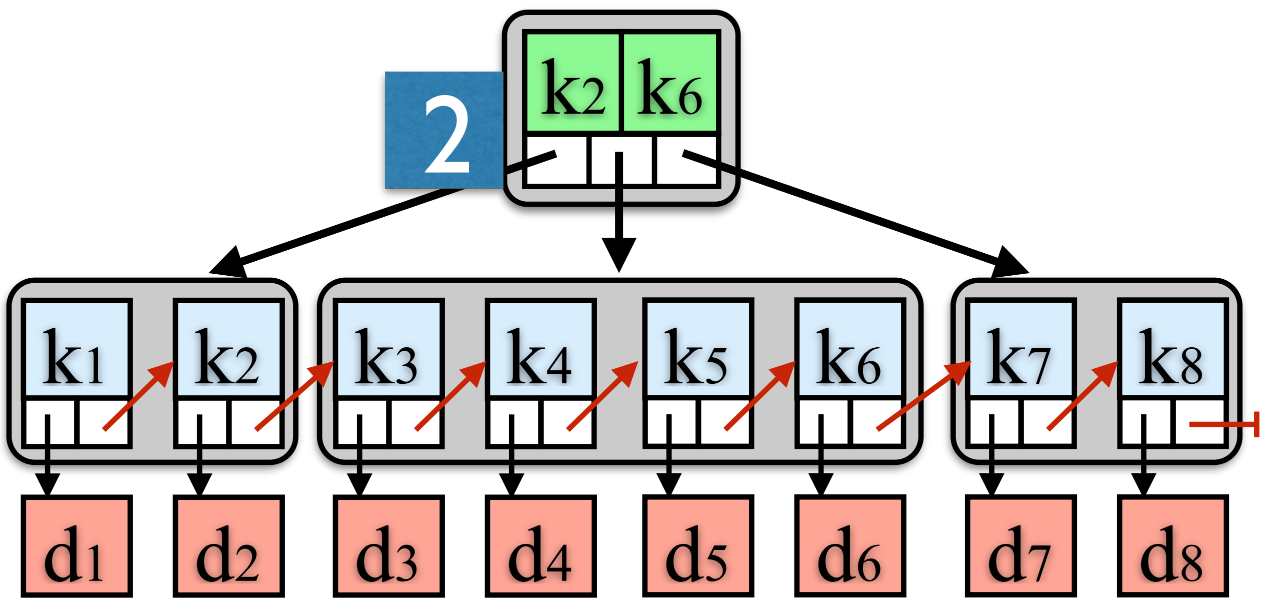
$I_B = I_{up} \cup I_{find} \cup I_{add} \cup I_{rem}$

B+ Tree

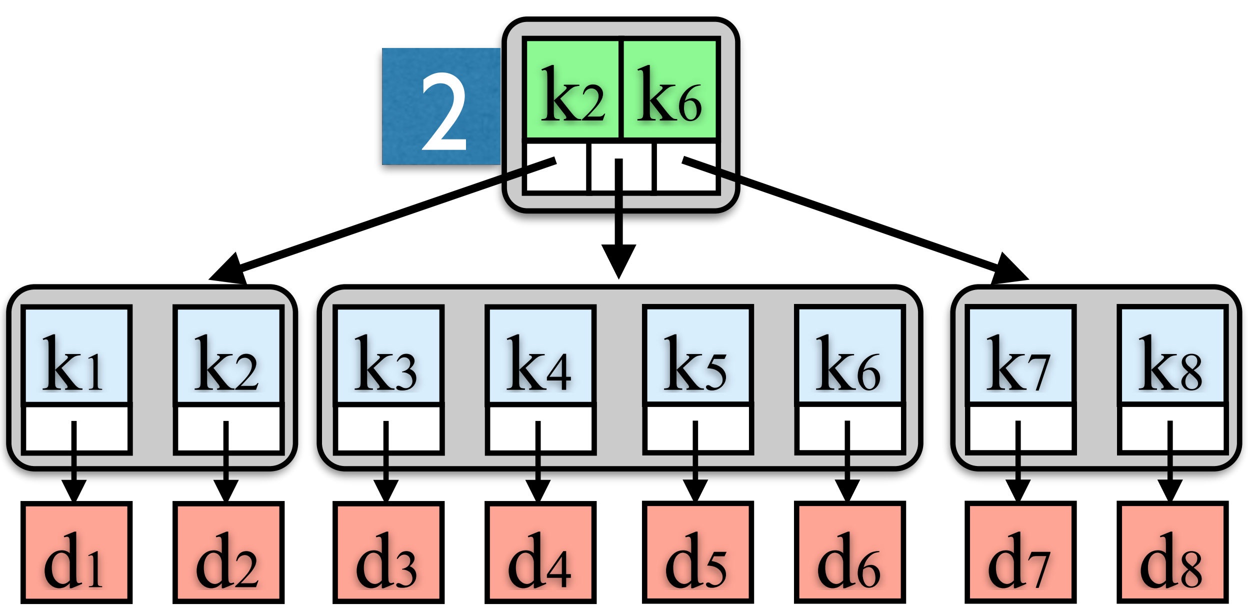


B+ Tree

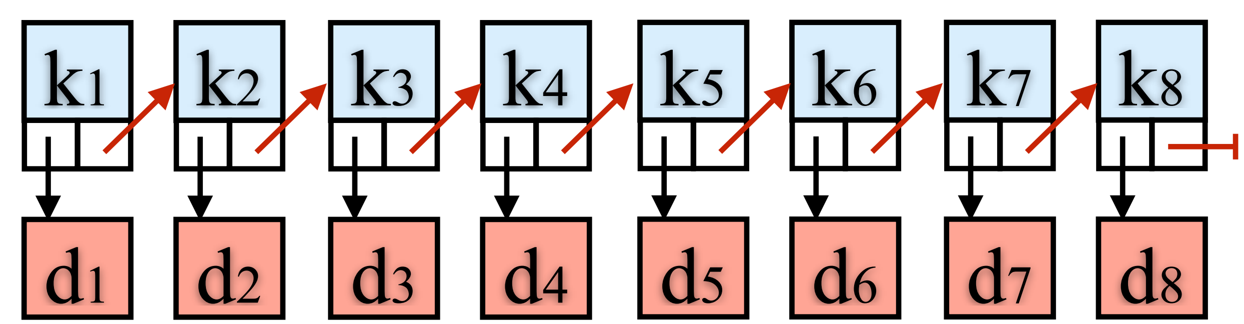
B+ Tree h L ⇔ BSTree h L ⊎* List L



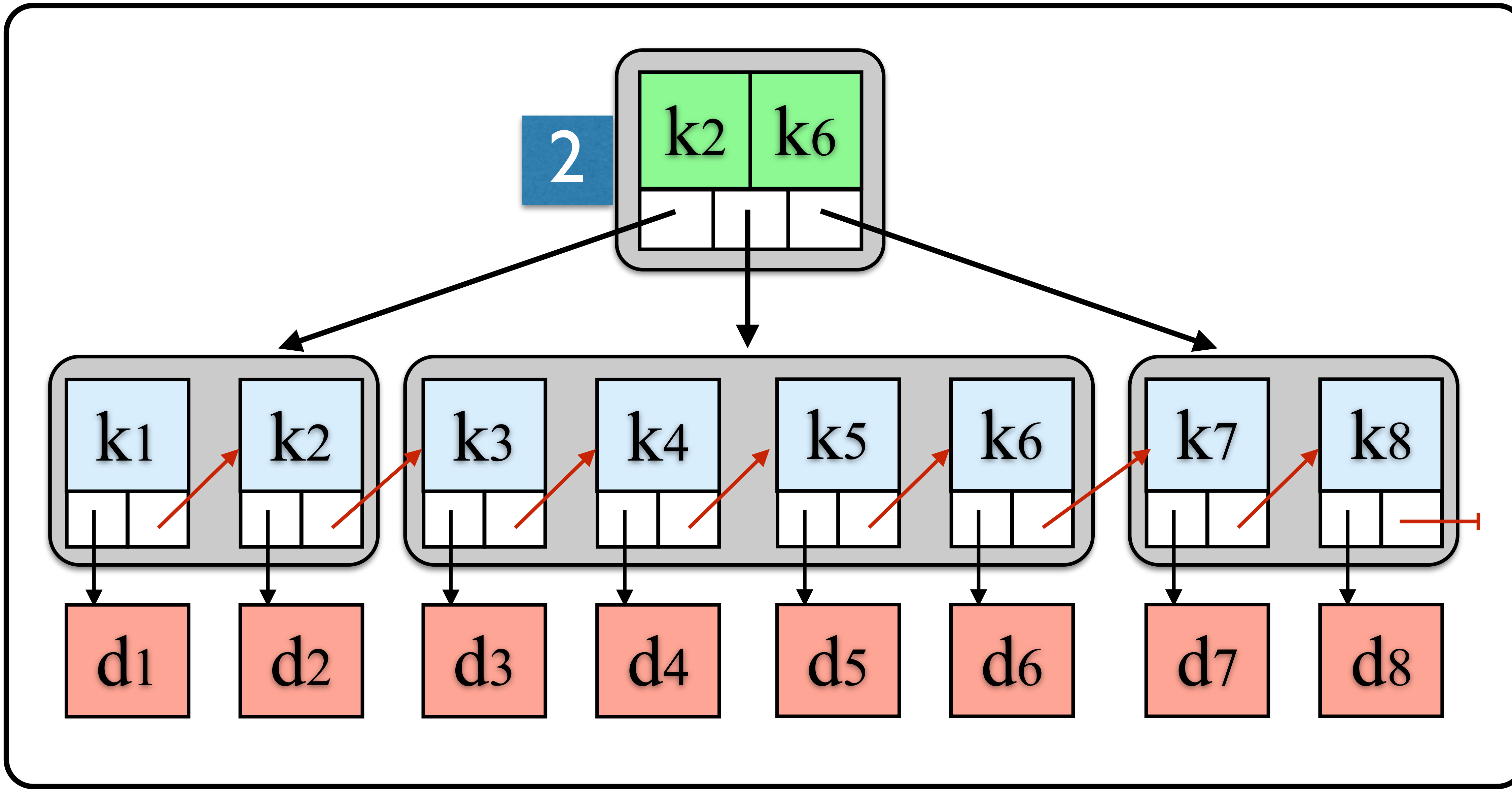
⇔



⊎*



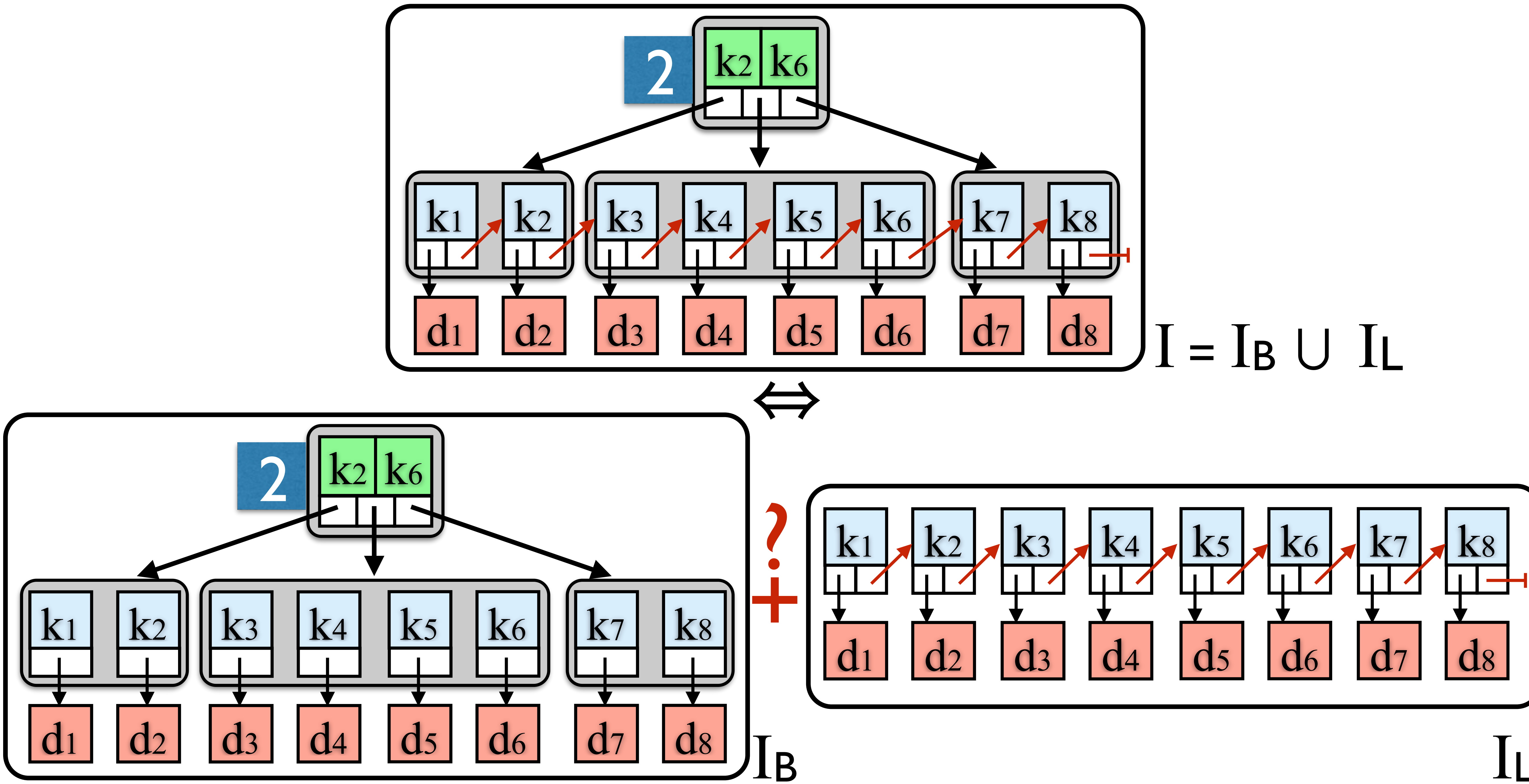
Concurrent B+ Tree



$I = I_B \cup I_L$

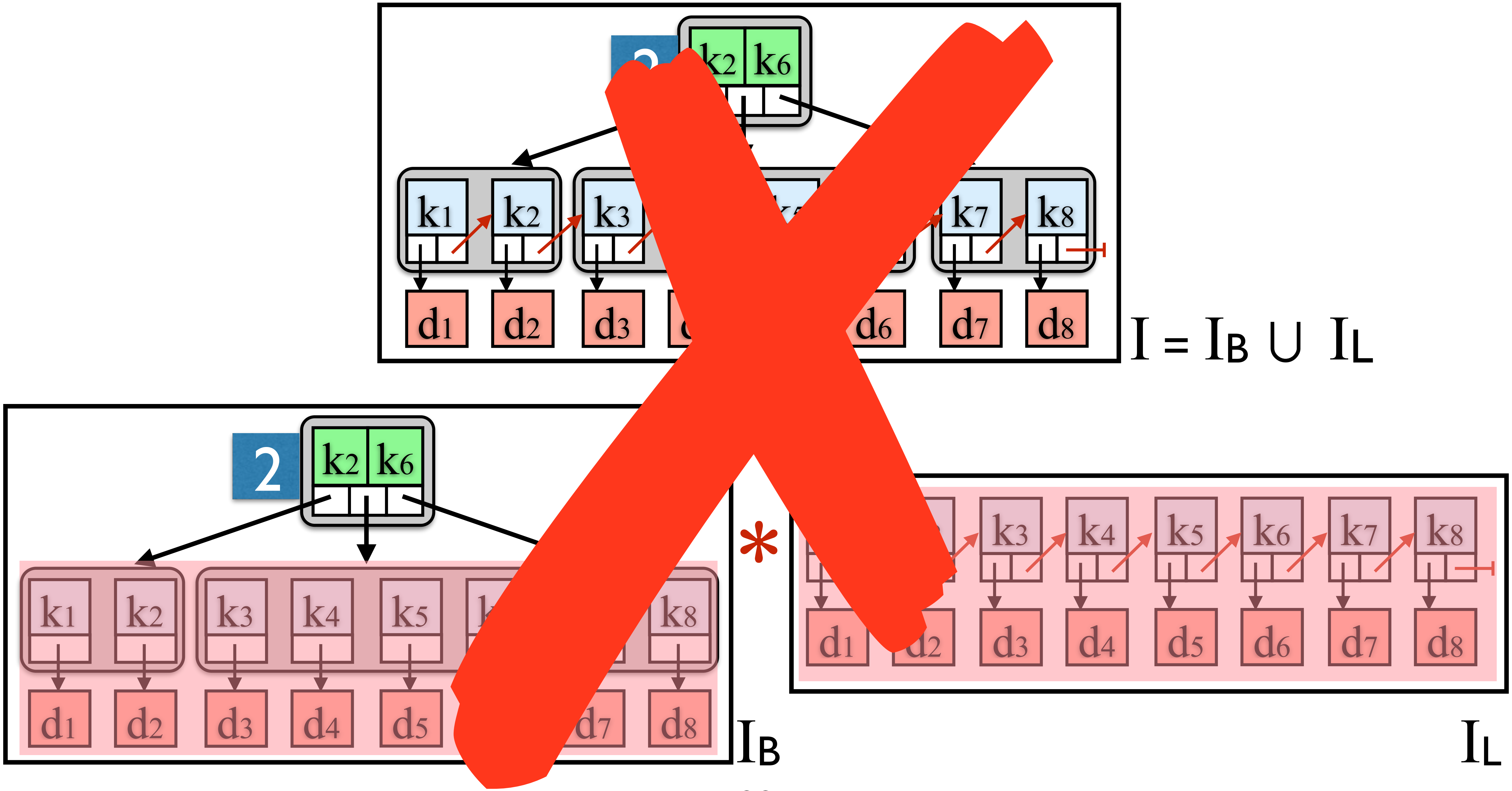
Concurrent B+ Tree Wish List

$$\boxed{\text{B+ Tree h L}}_{I_B \cup I_L} \Leftrightarrow \boxed{\text{BSTree h L}}_{I_B} \overset{?}{+} \boxed{\text{List L}}_{I_L}$$



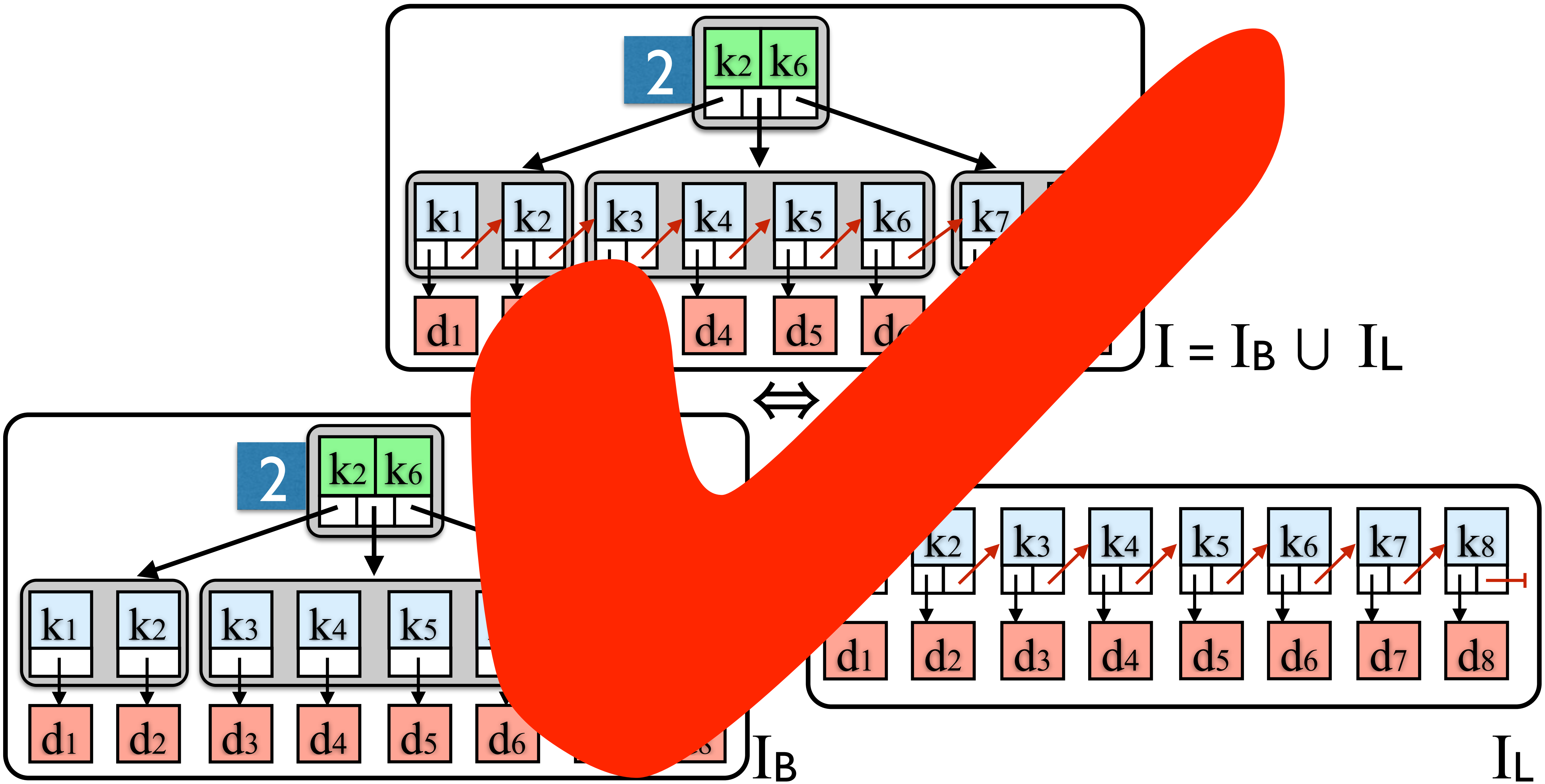
Concurrent B+ Tree
(Existing Approaches)

$\boxed{\text{B+ Tree h L}}_{I_B \cup I_L} \Leftrightarrow \boxed{\text{BSTree h L}}_{I_B} * \boxed{\text{List L}}_{I_L}$



Concurrent B+ Tree
(CoLoSL)

$$\boxed{\text{B+ Tree } h\ L}_{I_B \cup I_L} \Leftrightarrow \boxed{\text{BSTree } h\ L}_{I_B} \uplus^{*} \boxed{\text{List } L}_{I_L}$$



B+ Tree Wish List

- **Module Composition**

$$\boxed{\text{B+ Tree h L}}_{I_B \cup I_L} \Leftrightarrow \boxed{\text{BTree h L}}_{I_B} \mathbin{\uplus\!\!\!\!\!\ast} \boxed{\text{List L}}_{I_L}$$

- **Proof Modularity (overlapping frames)**
  - **Reuse** list-only operation (e.g. map)

$$\boxed{\text{BTree h L}}_{I_B} \mathbin{\uplus\!\!\!\!\!\ast} \boxed{\text{List L}}_{I_L} \xRightarrow{\text{(frame)}} \boxed{\text{List L}}_{I_L}$$

$$\left\{\boxed{\text{List L}}_{I_L}\right\}\ \texttt{inc\_all()}\ \left\{\boxed{\text{List inc(L)}}_{I_L}\right\}$$

  - **Reuse** tree-only operation (e.g. search)

$$\boxed{\text{BTree h L}}_{I_B} \mathbin{\uplus\!\!\!\!\!\ast} \boxed{\text{List L}}_{I_L} \xRightarrow{\text{(frame)}} \boxed{\text{BTree h L}}_{I_B}$$

$$\left\{\boxed{\text{BTree h L}}_{I_B}\right\}\ \texttt{search(k)}\ \left\{\boxed{\text{BTree h L}}_{I_B}\right\}$$

  - **Combine** tree/list module operations to implement B+ tree operations (e.g. remove)

CoLoSL Principles

$$\boxed{\text{B+ Tree h L}}_{I_B \cup I_L} \Leftrightarrow \boxed{\text{BSTree h L}}_{I_B} \mathbin{\uplus\!\!\!\!\ast} \boxed{\text{List L}}_{I_L}$$

$$\Downarrow$$

$$\boxed{\text{BSTree h L} \mathbin{\uplus\!\!\!\!\ast} \text{List L}}_{I_B \cup I_L}$$

Duplicating Resources

$$\boxed{P}_{I} \Rightarrow \boxed{P}_{I} \uplus \boxed{P}_{I}$$

CoLoSL Principles

$$\boxed{\text{B+ Tree h L}}_{I}$$

$$\Downarrow$$

$$\boxed{\text{BSTree h L} \uplus\!\!\!\!\ast \text{ List L}}_{I_B \cup I_L}$$

$$\Downarrow \qquad \text{(Copy)}$$

$$\boxed{\text{BSTree h L} \uplus\!\!\!\!\ast \text{ List L}}_{I_B \cup I_L} \uplus\!\!\!\!\ast \boxed{\text{BSTree h L} \uplus\!\!\!\!\ast \text{ List L}}_{I_B \cup I_L}$$

$$\boxed{\text{BSTree h L}}_{I_B} \uplus\!\!\!\!\ast \boxed{\text{List L}}_{I_L}$$

Forgetting Resources

$$\boxed{P \mathbin{\cup\!\!\!\!\!\ast\;} Q}_{I} \Rightarrow \boxed{P}_{I}$$

CoLoSL Principles

$$\boxed{\text{B+ Tree h L}}_{I}$$

$$\Downarrow$$

$$\boxed{\text{BTree h L} \mathbin{\cup\!\!\!\!*} \text{List L}}_{I}$$

$$\Downarrow$$

(Copy)

$$\boxed{\text{BSTree h L} \mathbin{\cup\!\!\!\!*} \text{List L}}_{I_B \cup I_L} \mathbin{\cup\!\!\!\!*} \boxed{\text{BSTree h L} \mathbin{\cup\!\!\!\!*} \text{List L}}_{I_B \cup I_L}$$

$$\Downarrow$$

(Forget)

$$\boxed{\text{BSTree h L}}_{I_B \cup I_L} \mathbin{\cup\!\!\!\!*} \boxed{\text{List L}}_{I_B \cup I_L}$$

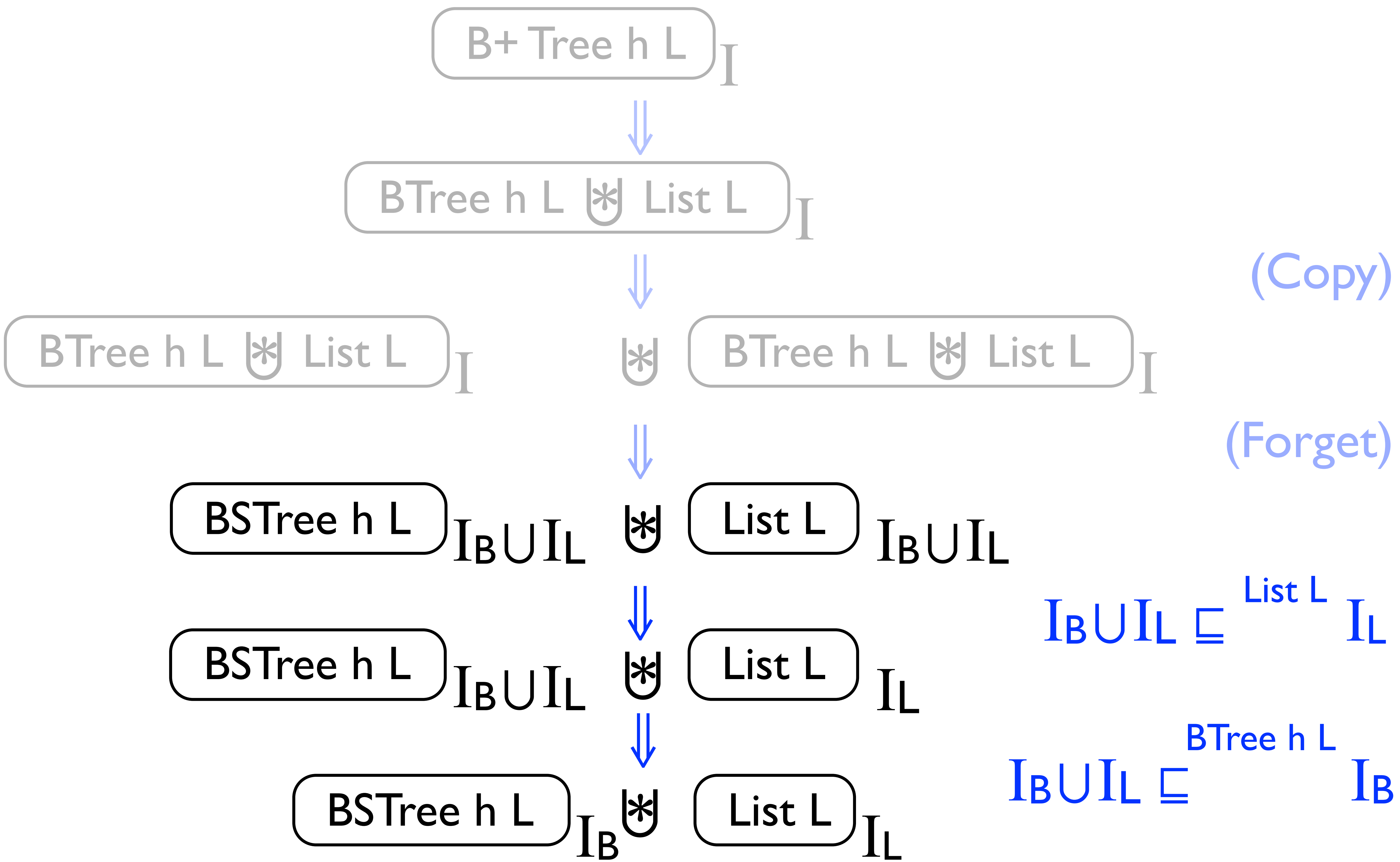
$$\boxed{\text{BTree h L}}_{I_B} \mathbin{\cup\!\!\!\!*} \boxed{\text{List L}}_{I_L}$$

Forgetting Interference (Shift)

if $\quad I \sqsubseteq^{P} I'$

then $\quad \boxed{P}_{I} \Rightarrow \boxed{P}_{I'}$

CoLoSL Principles



CoLoSL Principles

$$\boxed{\text{B+ Tree h L}}_{I_B \cup I_L} \Rightarrow \boxed{\text{BSTree h L}}_{I_B} \uplus\!\!\!\!* \; \boxed{\text{List L}}_{I_L}$$

$$\boxed{\text{B+ Tree h L}}_{I_B \cup I_L} \Leftarrow \boxed{\text{BSTree h L}}_{I_B} \uplus\!\!\!\!* \; \boxed{\text{List L}}_{I_L}$$

Merging Resources

$$\boxed{\;P\;}_{I} \mathbin{\cup\!\!\!\!\!*} \boxed{\;Q\;}_{I'} \Rightarrow \boxed{\;P \mathbin{\cup\!\!\!\!\!*} Q\;}_{I \cup I'}$$

CoLoSL Principles

$$\boxed{\text{BSTree h L}}_{I_B} \uplus\!\!\!\!*\; \boxed{\text{List L}}_{I_L}$$

$$\Downarrow \qquad \text{(Merge)}$$

$$\boxed{\text{BSTree h L} \uplus\!\!\!\!* \text{ List L}}_{I_B \cup I_L}$$

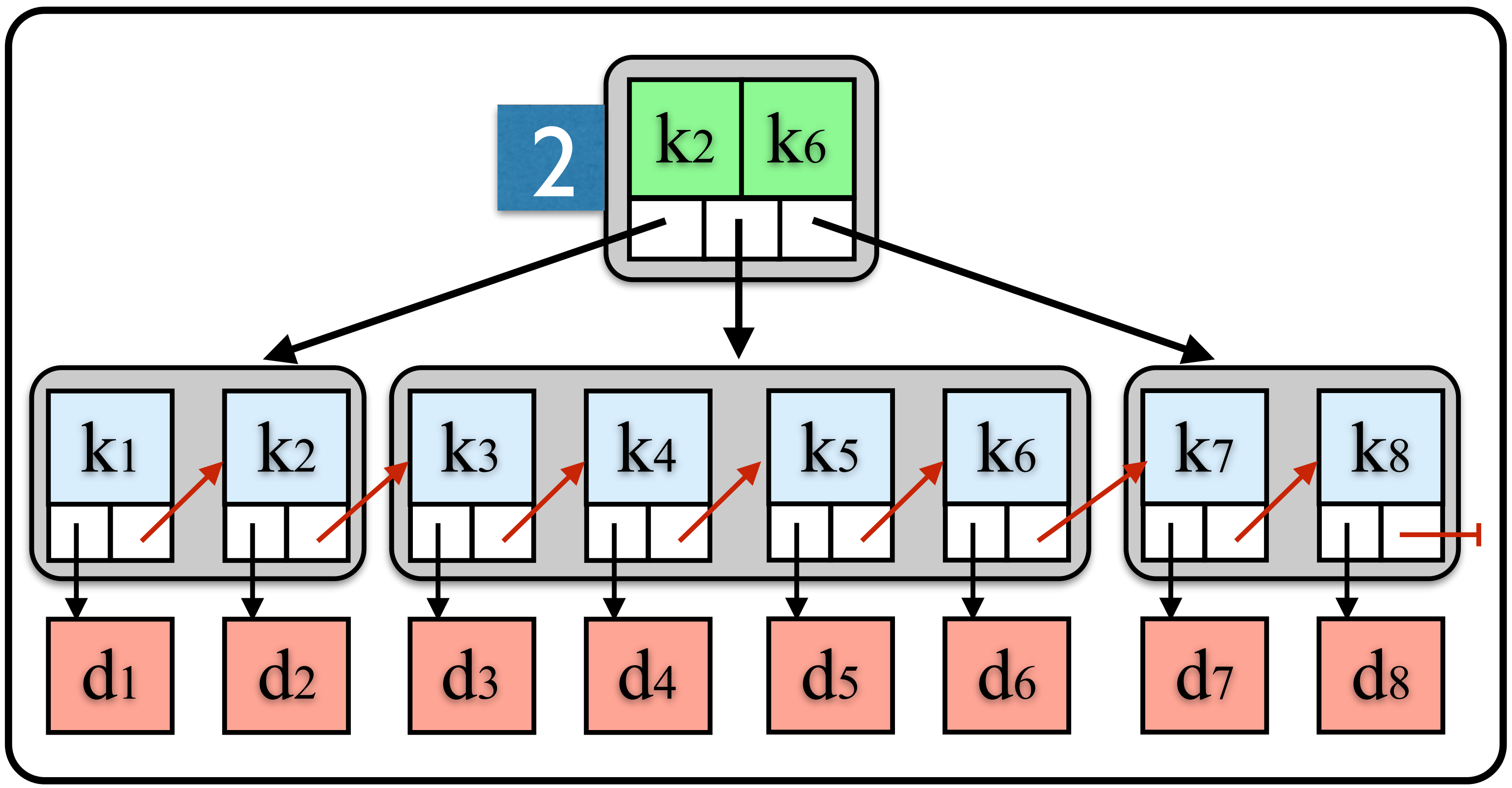
$$\Downarrow$$

$$\boxed{\text{B+ Tree h L}}_{I_B \cup I_L}$$

Examples

- **B+ Tree**
  - Module composition; proof modularity; better abstraction
  - Proof reuse - local proofs done in the largest context possible

B+ Tree: Possible Extension



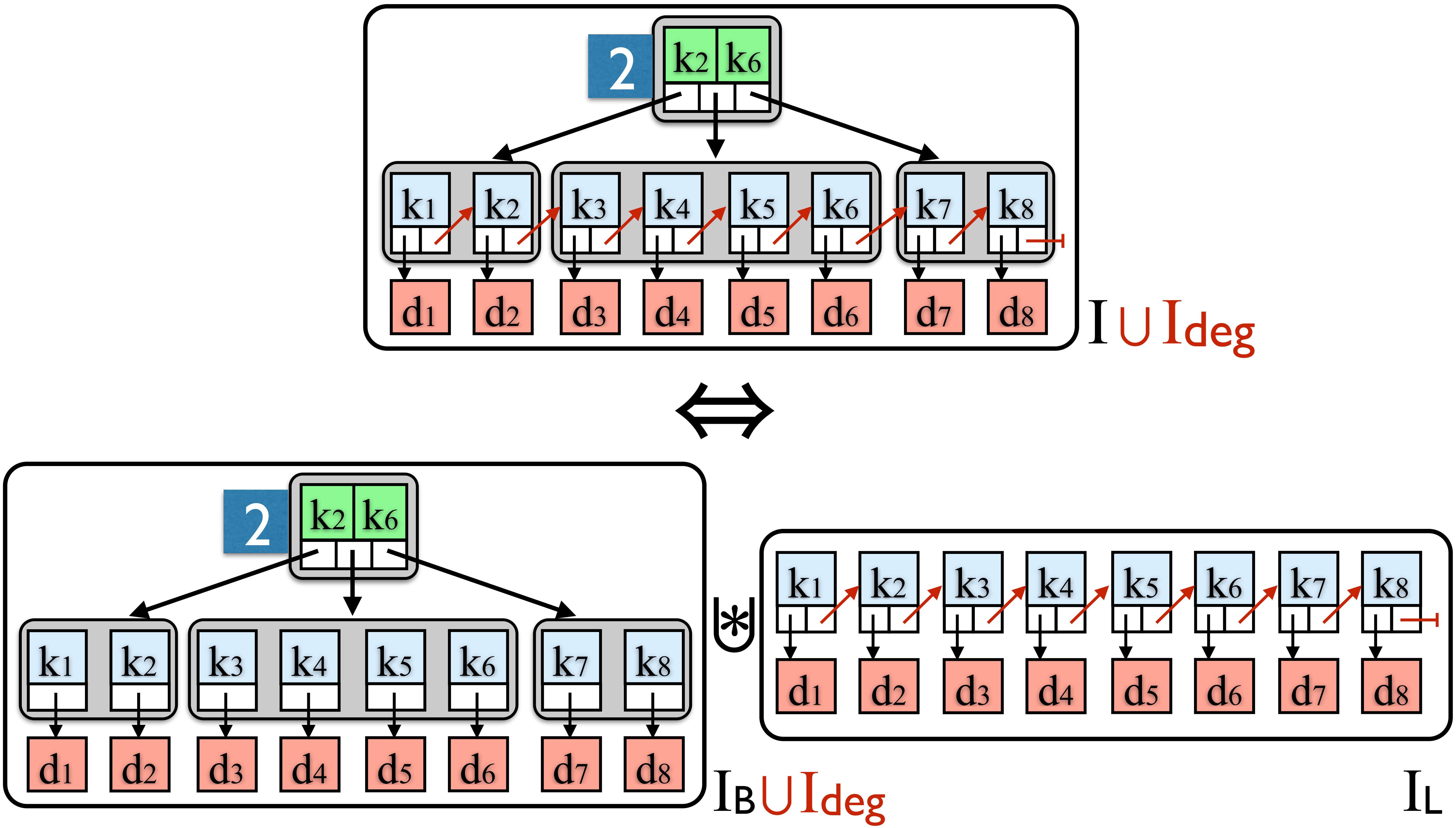
$I \cup I_{deg}$

- Re-degree the tree
  - periodically change the degree (d) for better search time
- Re-degreeing ONLY affects the tree structure (e.g. 2 ⇝ 3)

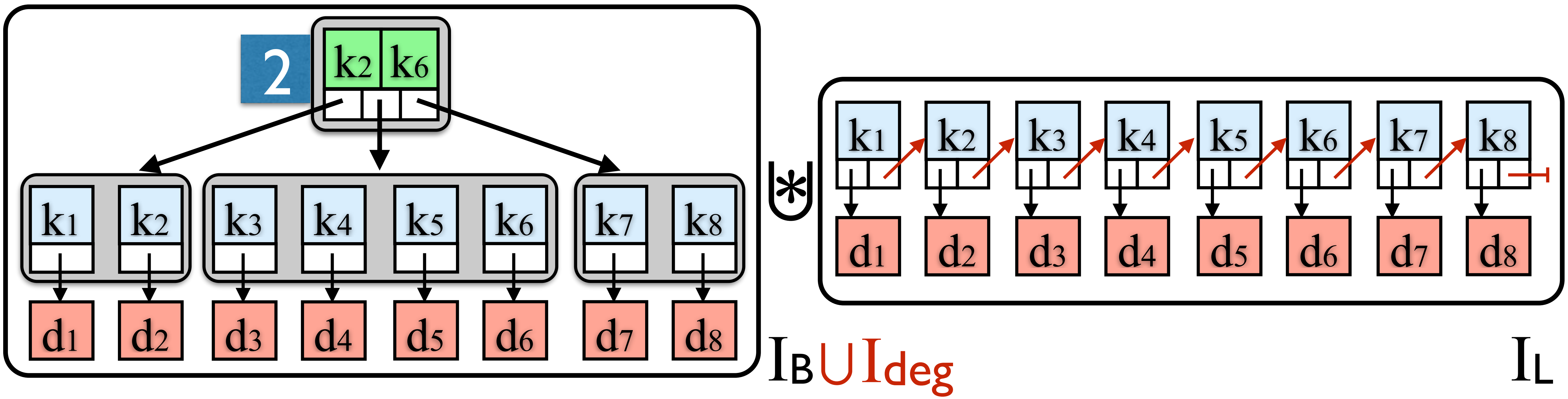
I_{deg} : d ⇝ d'

- Re-degreeing does NOT affect the list structure
  - Should not have to reprove list operations

B+ Tree: Possible Extension
(CoLoSL)



B+ Tree: Possible Extension
(CoLoSL)



- Re-degreeing extension only affects the tree
- Re-degreeing extension does NOT affect the list proofs
  - Can reuse the proofs as before

Examples

- B+ Tree
  - Module composition; proof modularity; better abstraction
  - Proof reuse
- Concurrent List
  - Dynamic extension
- Spanning Tree
  - Recursive overlapping graph predicate
  - Local proof; proof structures matches the algorithm
- Dijkstra’s Self-stabilising Token Ring
  - Local proof; proof Reuse

Conclusions

- From OG/Rg to CAP/TaDA
  - Huge steps towards compositionality/locality
  - Are we there yet? No!
- CoLoSL
  - Subjective/overlapping views
  - Flexible framing on shared resource/interference
  - Dynamic extension
  - More modular; better proof reuse
  - Are we there yet? Still No!
    - Abstraction layers; abstract atomicity, …

Thank you for listening!