Reasoning about B+ Trees with Operational Semantics and Separation Logic

Hayo Thielecke
(joint work with Alan Sexton)

University of Birmingham, UK

Introduction

B+ tree

- Balanced ordered search tree, $O(\log n)$ operations
- Widely used for efficient search in databases

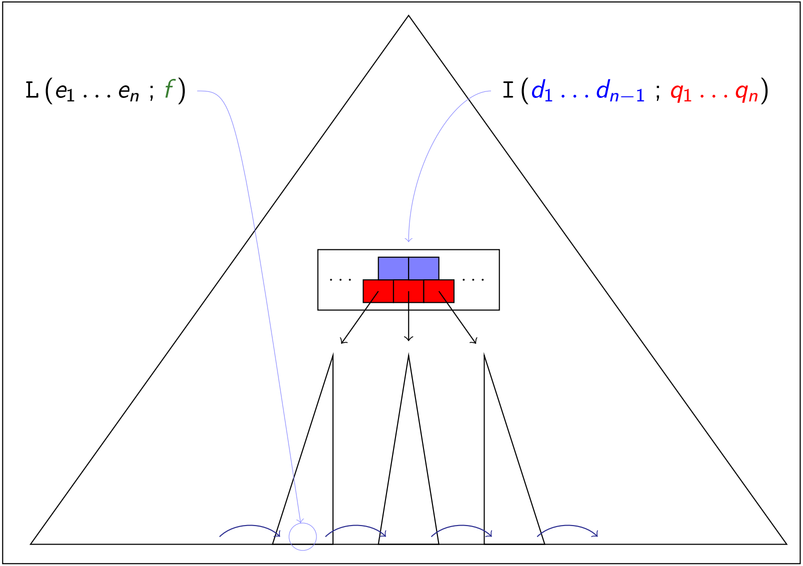
Operational semantics: Abstract Machine

- Formalizes operations on B+ trees, abstractly
- In the style of SECD machine or Krivine's machine

Separation logic

- Very expressive for inductive data structures with pointers
- Here: ordinary predicate logic extended with $*$ and $\mapsto$
- No Hoare logic $\{P\}\ c\ \{Q\}$

B+ tree visually



Operational semantics: abstract machine

Landin's SECD machine (1964)

Configurations: Stack, Environment, Control, Dump

$$\langle s \,,\, e \,,\, c \,,\, d \rangle \leadsto \langle s' \,,\, e' \,,\, c' \,,\, d' \rangle$$

Transitions by pattern matching (e.g., stack or control empty).

Our machine: insertion transitions

$$\langle \mathtt{C} \,,\, r \,,\, \pi \,,\, \sigma \rangle \leadsto \ldots$$

$\mathtt{C}$: command, e.g., $\mathtt{Insert}\,(e)$, $\mathtt{S}$, $\mathtt{D}\,(k, q)$

r: root of current subtree

π: stack, used for tree traversal

σ: store, holds nodes of the tree, $\approx$ heap in Sep Logic

Separation logic connectives

$$
\begin{array}{lcl}
\sigma & \models & p \mapsto N \text{ iff } \sigma = \{p \mapsto N\} \text{ for a node } N \text{ (leaf or internal)} \\
\sigma & \models & Q_0 * Q_1 \text{ iff } \sigma = \sigma_0 \cup \sigma_1 \text{ where } \sigma_0 \cap \sigma_1 = \emptyset \\
 & & \text{ and } \sigma_0 \models Q_0 \text{ and } \sigma_1 \models Q_1 \\
\sigma & \models & Q_0 \wedge Q_1 \text{ iff } \sigma \models Q_0 \text{ and } \sigma \models Q_1 \\
\sigma & \models & Q_0 \vee Q_1 \text{ iff } \sigma \models Q_0 \text{ or } \sigma \models Q_1 \\
\sigma & \models & \mathtt{true} \text{ iff } \sigma \text{ is any store} \\
\sigma & \models & \mathtt{emp} \text{ iff } \mathrm{dom}(\sigma) = \emptyset
\end{array}
$$

Local reasoning: footprint and frame:

$$
\frac{\{P\}\ c\ \{Q\}}{\{P * R\}\ c\ \{Q * R\}}
$$

B+ trees in Separation Logic

Predicate describes data structure

$\sigma \models \text{Btree}_h(r, S, a, z, n)$

- height h
- r points to root node of tree
- set of entries S
- first leaf at address a, last leaf node points to z
- size of root is at least n

Induction on height: Leaf case

$$\begin{array}{rcl} \text{Btree}_1(r, S, a, z, n) & \iff & \exists e_1, \ldots, e_n.\ n \leqslant MaxN \\ & \wedge & r \mapsto \mathtt{L}\,(e_1 \ldots e_n\ ;\ z) \\ & \wedge & S = \{e_1, \ldots, e_n\} \wedge a = r \wedge e_1 \sqsubset \cdots \sqsubset e_n \end{array}$$

B+ tree continued: internal nodes

$$
\begin{array}{rcl}
\mathrm{Btree}_{h+1}(r, S, a, z, n) & \iff & \exists d_1, \ldots, d_{n-1}, q_1 \ldots q_n, m_1, \ldots, m_n. \\
 & \wedge & (r \mapsto \mathtt{I}\,(d_1 \ldots d_{n-1}\;;\;q_1 \ldots q_n) \\
 & & \;\ast\; \mathrm{Btree}_h(q_1, S_1, a_1, a_2, m_1) \\
 & & \;\ast\; \mathrm{Btree}_h(q_2, S_2, a_2, a_3, m_2) \\
 & & \;\ast\; \cdots \\
 & & \;\ast\; \mathrm{Btree}_h(q_n, S_n, a_n, a_{n+1}, m_n)) \\
 & \wedge & a_1 = a \wedge a_{n+1} = z \\
 & \wedge & S = S_1 \cup \cdots \cup S_n \\
 & \wedge & (\forall j.\ 1 < j < n - 1 \Rightarrow d_j \sqsubseteq S_j \sqsubset d_{j+1}) \\
 & \wedge & (\forall j.\ 1 < j \leqslant n \Rightarrow \lceil MaxN/2 \rceil \leqslant m_j) \\
 & \wedge & (S_1 \sqsubset d_1) \\
 & \wedge & (d_{n-1} \sqsubseteq S_n)
\end{array}
$$

B+ tree insertion

- Invariant: balanced
- Insertion may split a node.
- Parent node may then split as well.
- Splitting ripples up the tree recursively.
- Insertion produces either a subtree or two subtrees.
- In the operational semantics: "single" or "double" commands

$$\left\langle \mathtt{S}\,,\, r\,,\, \pi\,,\, \sigma \right\rangle$$

$$\left\langle \mathtt{D}\,(k, q)\,,\, r\,,\, \pi\,,\, \sigma \right\rangle$$

Insert Rules — Descending the tree and splitting leaf

$$\langle \texttt{Insert}\,(a)\,,\, r\,,\, \pi\,,\, \sigma \rangle \quad \rightsquigarrow \quad \langle \texttt{Insert}\,(a)\,,\, \mathbf{p}_i\,,\, (r, i) :: \pi\,,\, \sigma \rangle$$

if $\sigma(r) = \texttt{I}\,(\mathbf{d}\ ;\ \mathbf{p})$

where $i = \texttt{first}\,(\mathbf{d}, \lambda x.\ x > \texttt{key}\,(a))$

$$\langle \texttt{Insert}\,(a)\,,\, r\,,\, \pi\,,\, \sigma \rangle \quad \rightsquigarrow \quad \left\langle \texttt{D}\,(k, q)\,,\, r\,,\, \pi\,,\, \sigma \begin{bmatrix} r \mapsto \texttt{L}\,(\mathbf{e}'\ ;\ q)\,, \\ q \mapsto \texttt{L}\,(\mathbf{e}''\ ;\ f) \end{bmatrix} \right\rangle$$

where $\sigma(r) = \texttt{L}\,(\mathbf{e}\ ;\ f)$

and $i = \texttt{first}\,(\mathbf{e}, \lambda x.\ \texttt{key}\,(x) \geqslant \texttt{key}\,(a))$

and $\langle \mathbf{e}', k, \mathbf{e}'' \rangle = \texttt{splitL}\,(i, a, \mathbf{e})$

and $q \notin \mathrm{dom}(\sigma)$

Insert Rules — Returning up the tree with $\mathtt{S}$ or $\mathtt{D}\,(k, q)$

$$\langle \mathtt{S}\,,\, r\,,\, (t, i) :: \pi\,,\, \sigma \rangle \;\leadsto\; \langle \mathtt{S}\,,\, t\,,\, \pi\,,\, \sigma \rangle$$

$$\langle \mathtt{D}\,(k, q)\,,\, r\,,\, (t, i) :: \pi\,,\, \sigma \rangle \;\leadsto\; \langle \mathtt{S}\,,\, t\,,\, \pi\,,\, \sigma[t \mapsto \mathtt{I}\,(\mathbf{d}' \,;\, \mathbf{p}')] \rangle$$

if $|\mathbf{p}| < \mathit{MaxN}$

where $\sigma(t) = \mathtt{I}\,(\mathbf{d} \,;\, \mathbf{p})$

and $\mathbf{d}' = \mathtt{ins}\,(k, i, \mathbf{d})$

and $\mathbf{p}' = \mathtt{ins}\,(q, i+1, \mathbf{p})$

$$\langle \mathtt{D}\,(k, q)\,,\, r\,,\, (t, i) :: \pi\,,\, \sigma \rangle \;\leadsto\; \left\langle \mathtt{D}\,(k', q')\,,\, t\,,\, \pi\,,\, \sigma \begin{bmatrix} t \mapsto \mathtt{I}\,(\mathbf{d}' \,;\, \mathbf{p}')\,, \\ q' \mapsto \mathtt{I}\,(\mathbf{d}'' \,;\, \mathbf{p}'') \end{bmatrix} \right\rangle$$

where $\sigma(t) = \mathtt{I}\,(\mathbf{d} \,;\, \mathbf{p})$

and $\langle \mathbf{d}', \mathbf{p}', k', \mathbf{d}'', \mathbf{p}'' \rangle = \mathtt{splitI}\,(i, k, q, \mathbf{d}, \mathbf{p})$

and $q' \notin \mathrm{dom}(\sigma)$

Correctness of insertion Idea: *footprint* $*$ *frame* for the same stack π.

Lemma

Assume $\sigma \models \mathrm{Btree}_h(r, S, a, z, n) * R$. *Then either:*

1. $\langle \mathtt{Insert}\,(e)\,,\, r\,,\, \pi\,,\, \sigma \rangle \rightsquigarrow \cdots \rightsquigarrow \langle \mathtt{S}\,,\, r\,,\, \pi\,,\, \sigma' \rangle$ *and*

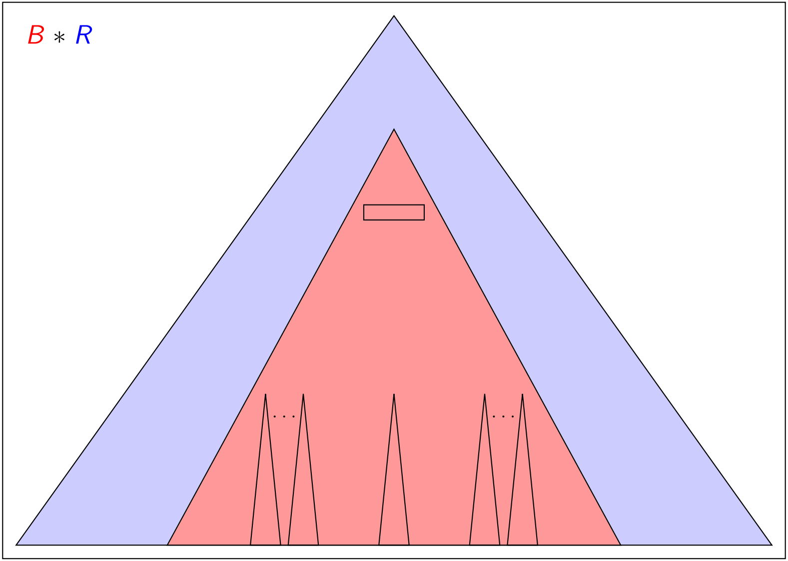
$$\sigma' \models \mathrm{Btree}_h(r, S + e, a, z, n) * R$$

2. $\langle \mathtt{Insert}\,(e)\,,\, r\,,\, \pi\,,\, \sigma \rangle \rightsquigarrow \cdots \rightsquigarrow \langle \mathtt{D}\,(k, q)\,,\, r\,,\, \pi\,,\, \sigma' \rangle$ *and*

$$\sigma' \models \mathrm{Btree}_h(r, S_r, a, b, n') * \mathrm{Btree}_h(q, S_q, b, z, n'') * R$$

Moreover, $S_r \cup S_q = S + e$ *and* $S_r \sqsubset k \sqsubseteq S_q$.

Proving Correctness of Insertion



Proving Correctness of Insertion

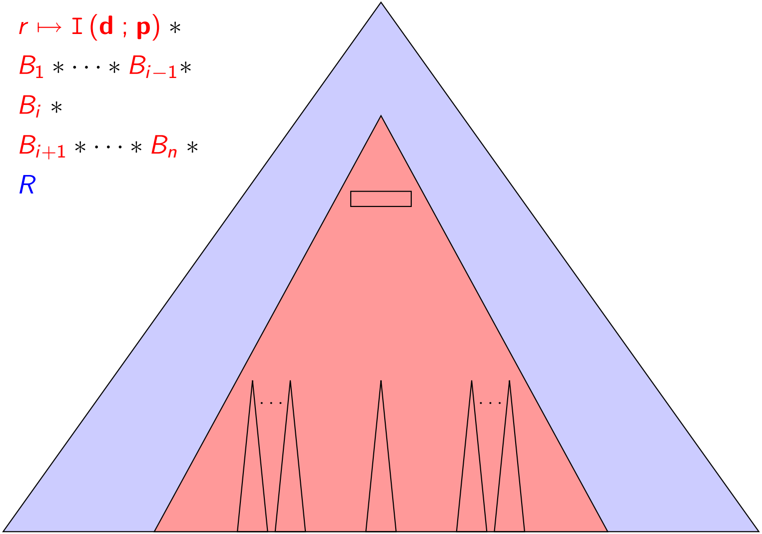
$r \mapsto \mathtt{I}\,(\mathbf{d}\;;\,\mathbf{p}) \,*$

$B_1 * \cdots * B_{i-1} *$

$B_i \,*$

$B_{i+1} * \cdots * B_n \,*$

R



Proving Correctness of Insertion

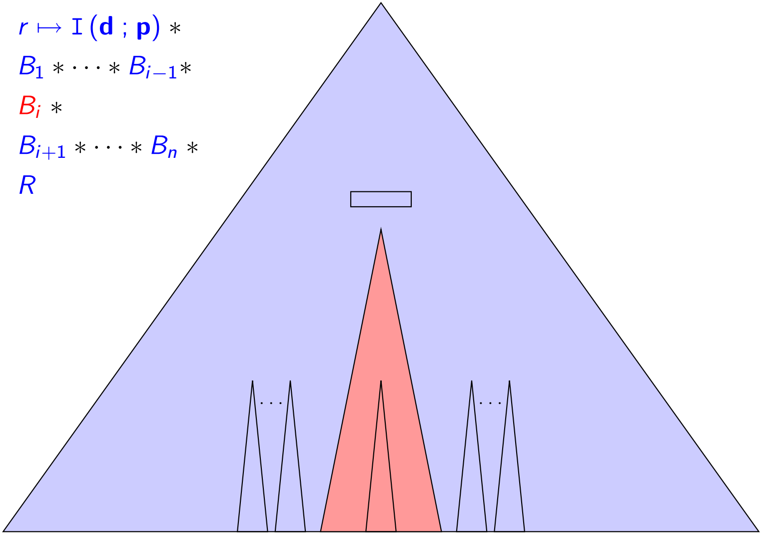
$r \mapsto \mathtt{I}\,(\mathbf{d}\ ;\ \mathbf{p}) \ *$

$B_1 * \cdots * B_{i-1} *$

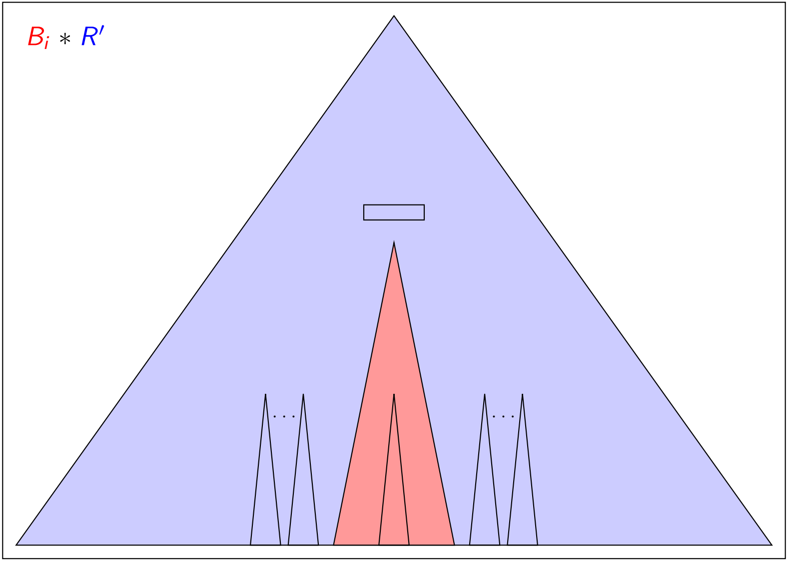
$B_i \ *$

$B_{i+1} * \cdots * B_n \ *$

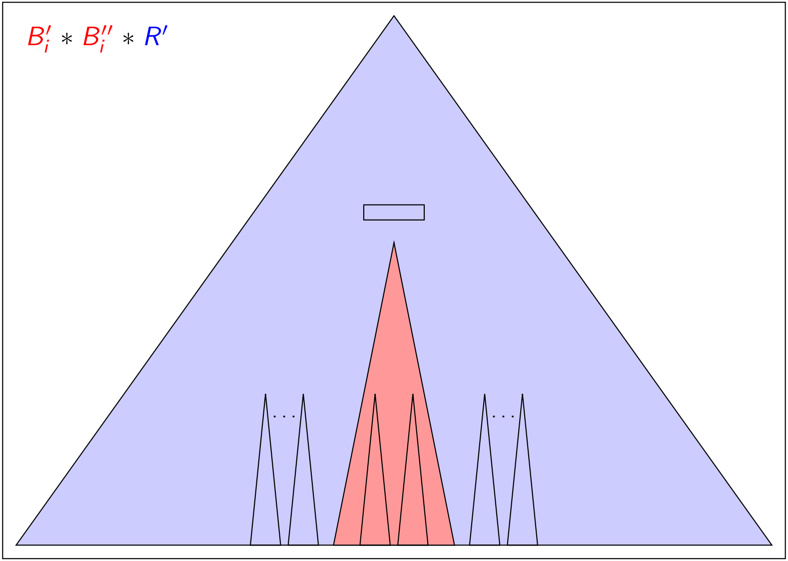
R



Proving Correctness of Insertion



Proving Correctness of Insertion



Proving Correctness of Insertion

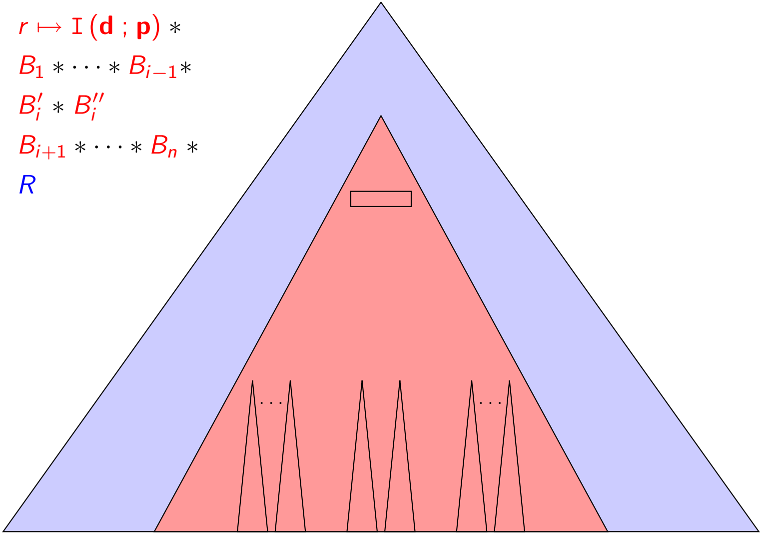
$r \mapsto \mathtt{I}\,(\mathbf{d}\ ;\ \mathbf{p}) \ *$

$B_1 * \cdots * B_{i-1} *$

$B_i' * B_i''$

$B_{i+1} * \cdots * B_n *$

R



Finding a list of entries for a range query

- Find the list of entries with key $\geqslant k$.
- Need to find the first such element at the leaf level
- Use fringe links of the leaf nodes
- Machine descends the tree (no need to keep a stack).

$$
\begin{array}{lcl}
\left\langle \mathtt{Find}\,(k)\,,\, r\,,\, \sigma \right\rangle & \rightsquigarrow & \left\langle \mathtt{Find}\,(k)\,,\, \mathbf{p}_i\,,\, \sigma \right\rangle \\
 & & \quad \text{if } \sigma(r) = \mathtt{I}\,(\mathbf{d}\ ;\ \mathbf{p}) \\
 & & \quad \text{where } i = \mathtt{first}\,(\mathbf{d}, \lambda x.\ x > k) \\
\left\langle \mathtt{Find}\,(k)\,,\, r\,,\, \sigma \right\rangle & \rightsquigarrow & \langle \mathtt{Ret}(r, i), \sigma \rangle \\
 & & \quad \text{where } \sigma(r) = \mathtt{L}\,(\mathbf{e}\ ;\ f) \\
 & & \quad \text{and } i = \mathtt{first}\,(\mathbf{e}, \lambda x.\ \mathtt{key}\,(x) \geqslant k)
\end{array}
$$

Correctness of find

- Start from a B+ tree with its implicit fringe list
- Result of find is covered by a list predicate
- Rest of the tree is swept into the catch-all predicate true.
- Gerrymander the existing store to reveal the result list.

Theorem

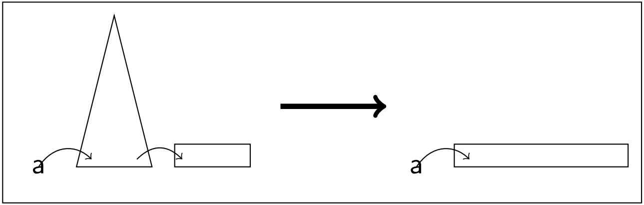
If $\sigma \models \exists h, a.\mathrm{Btree}_h(r, S, a, \mathtt{null}, n)$, *then*

$$\left\langle \mathtt{Find}\,(k)\,,\ r\,,\ \sigma \right\rangle \rightsquigarrow \cdots \rightsquigarrow \langle \mathtt{Ret}(q, i), \sigma \rangle$$

for some q *and* i *with*

$$\sigma \models \underbrace{\mathrm{FList}(q, i, S \uparrow k)}_{\textit{list of results}} \quad * \quad \underbrace{\mathtt{true}}_{\textit{everything else}}$$

Correctness of find Appending the fringe list onto an accumulator.



Lemma

Suppose

$$\sigma \models \mathrm{Btree}_h(r, S_r, a, z, n) * \mathrm{FList}(z, 1, S_z)$$

Then $\sigma \models \mathrm{FList}(a, 1, S_r \cup S_z) * \mathtt{true}$.

Correctness of find: induction

A technical lemma combines:

- current subtree
- partial result list
- everything else, by way of `true`

Lemma

Let

$$\sigma \models \mathsf{Btree}_h(p, S_p, a, z, n) * \mathsf{FList}(z, 1, S_z) * \mathtt{true}$$

Then we have a sequence of h transitions

$$\big\langle \mathtt{Find}\,(k)\,,\; r\;,\; \sigma \big\rangle \rightsquigarrow \cdots \rightsquigarrow \langle \mathtt{Ret}(q, i), \sigma\rangle$$

for some q and i with $\sigma \models \mathsf{FList}(q, i, (S_r \uparrow k) \cup S_z) * \mathtt{true}$.

Conclusions

Summary

- Programming semantics technology to address B+ trees
- Separation logic $\mapsto$ and $*$ for invariants and local reasoning
- Abstract machines from functional programming
- Translation to CAML prototype is straightforward

Further work

- Deletion will be covered in the full version of the paper
- High-level primitives, like Gardner's context logic for XML
- Trees with "holes", holey Brick and BV tree: a job for $-\!*$?
- Concurrent B trees

The End

Thank you for your attention.