Verifying Robustness of Programs Under Structural Perturbations

Clay Thomas and Jacob Bond

November 29, 2017

Motivation

- “Sanity Checks”

- Functions invariant under order of list
  - max, sum, sort

- Data structures with representing something else
- Invariance under value it represents
  - binary search trees, heaps, hash sets

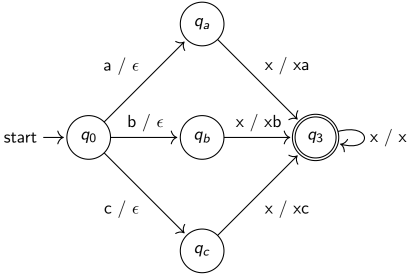
Lists – Invariance under order

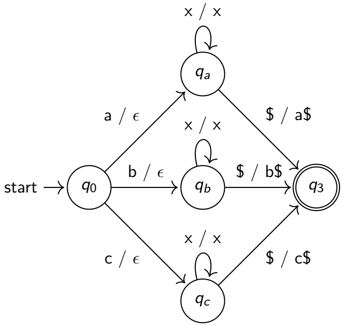
Given an array a

- Let a_{swap} be a with its first and second entry swapped
  - $[a[1], a[0], a[2], a[3], \ldots, a[n]]$
- Let a_{rot} be a rotated by 1
  - $[a[1], a[2], a[3], \ldots a[n], a[0]]$

Lemma: If for any a, $P(a) = P(a_{swap}) = P(a_{rot})$, then for any permutation a' of a, we have $P(a) = P(a')$.
Proof: Math

Automata – Invariance under order





Automata – Invariance under order

Theorem: Given a deterministic automata M, we can check if M is invariant under the order of its input in time $O(|\Sigma|^2|M|\log(|\Sigma||M|))$.

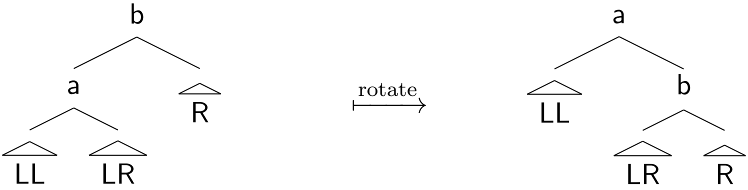
Proof:

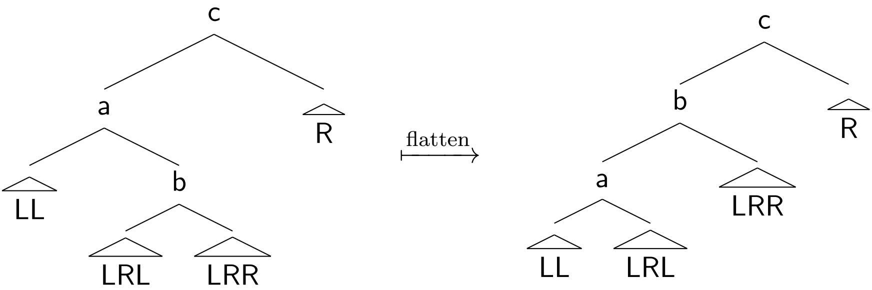
- Construct the machines M_{swap} and M_{rot} by composing M with those machines
  - Requires $O(|\Sigma||M|)$ states
- Check if $L(M_{swap}) = L(M) = L(M_{rot})$
  - E.g. by state minimization in time $O(n|\Sigma|\log n)$

Binary Search Trees

- For lists, two simple permutations generated all permutations
- Goal: similar permutations for BSTs

Binary Search Trees





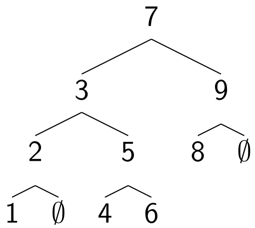
Binary Search Trees

It suffices to show

- Every tree can be transformed into a “normal form” (i.e. list)
- Every operation is reversable

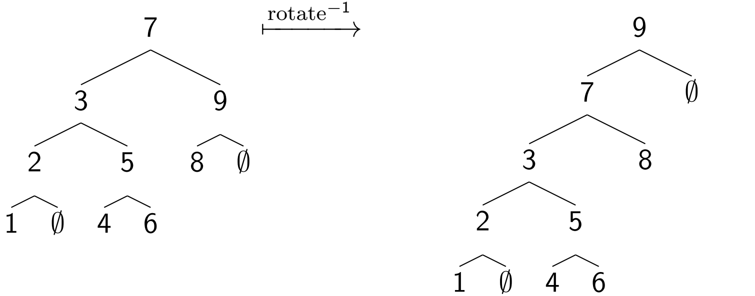
Binary Search Trees – Proof by example

- Every tree can be transformed into a “normal form” (i.e. list)



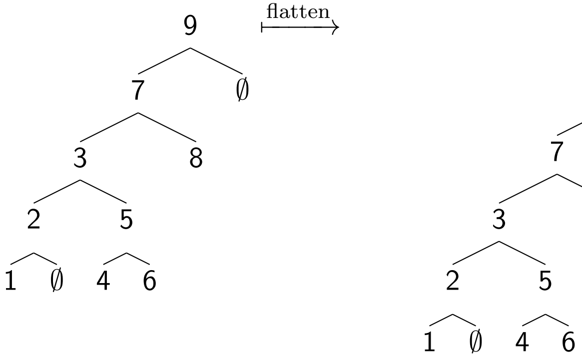
Binary Search Trees – Proof by example

- Every tree can be transformed into a “normal form” (i.e. list)



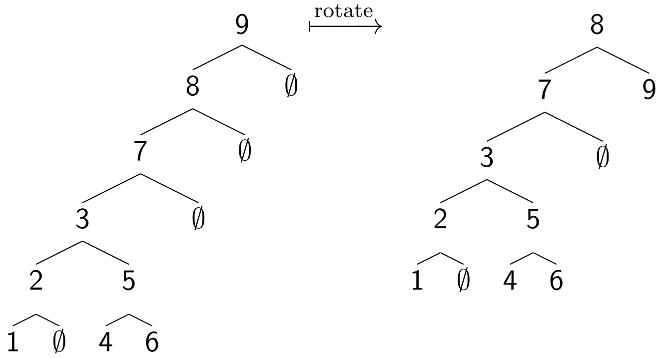
Binary Search Trees – Proof by example

- Every tree can be transformed into a “normal form” (i.e. list)



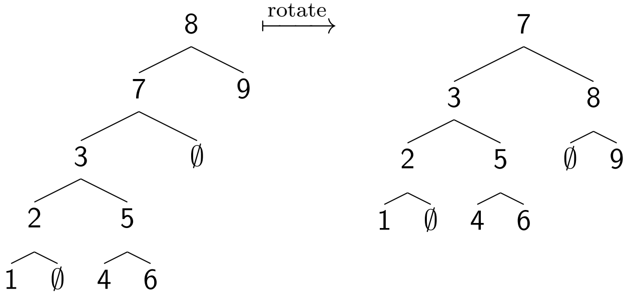
Binary Search Trees – Proof by example

- Every tree can be transformed into a “normal form” (i.e. list)



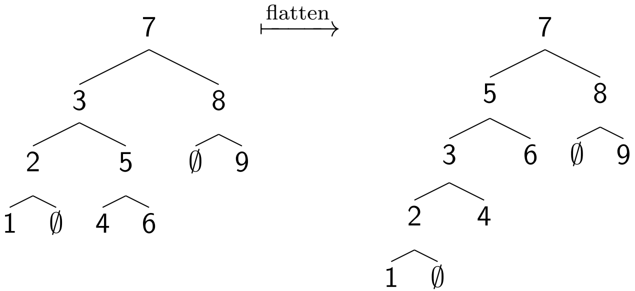
Binary Search Trees – Proof by example

- Every tree can be transformed into a “normal form” (i.e. list)



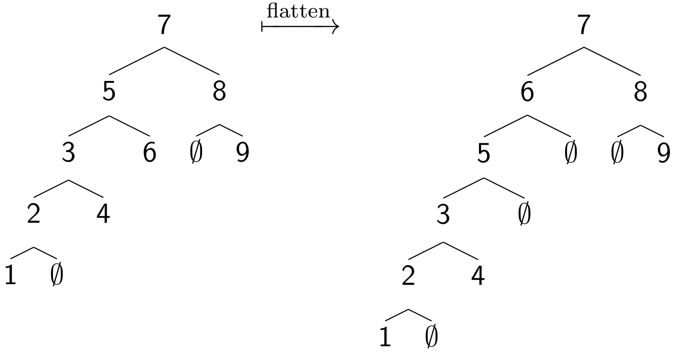
Binary Search Trees – Proof by example

- Every tree can be transformed into a “normal form” (i.e. list)



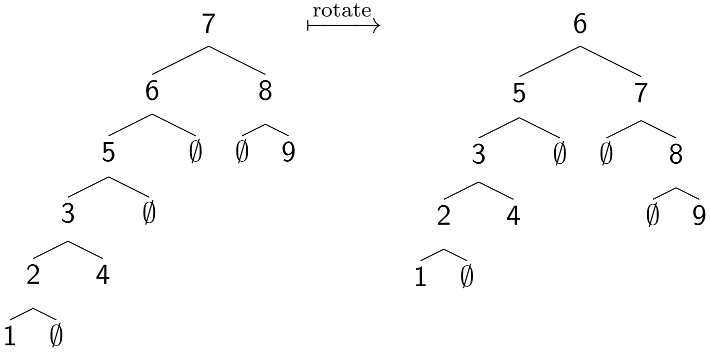
Binary Search Trees – Proof by example

- Every tree can be transformed into a "normal form" (i.e. list)



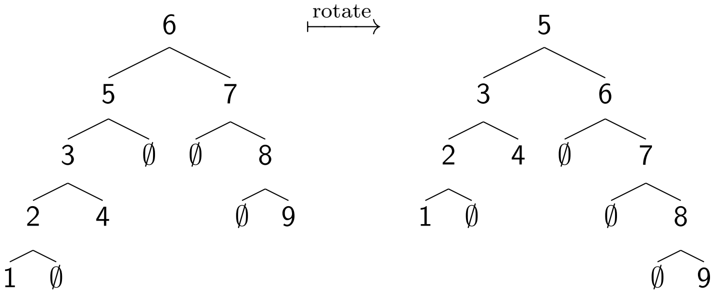
Binary Search Trees – Proof by example

- Every tree can be transformed into a "normal form" (i.e. list)



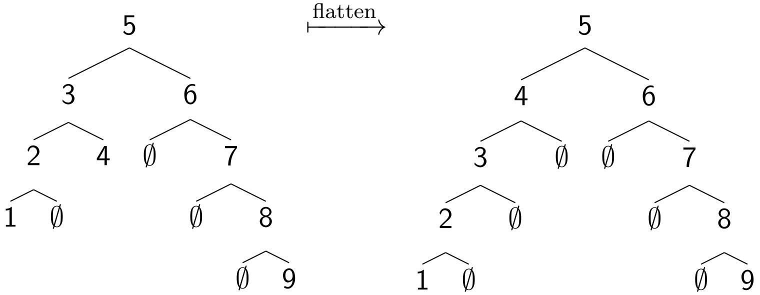
Binary Search Trees – Proof by example

- Every tree can be transformed into a “normal form” (i.e. list)



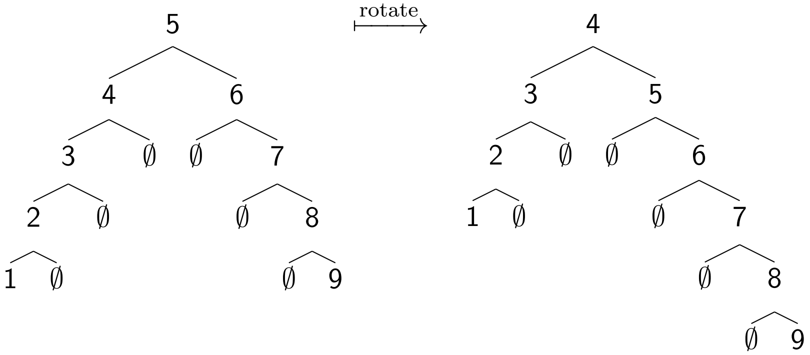
Binary Search Trees – Proof by example

- Every tree can be transformed into a "normal form" (i.e. list)



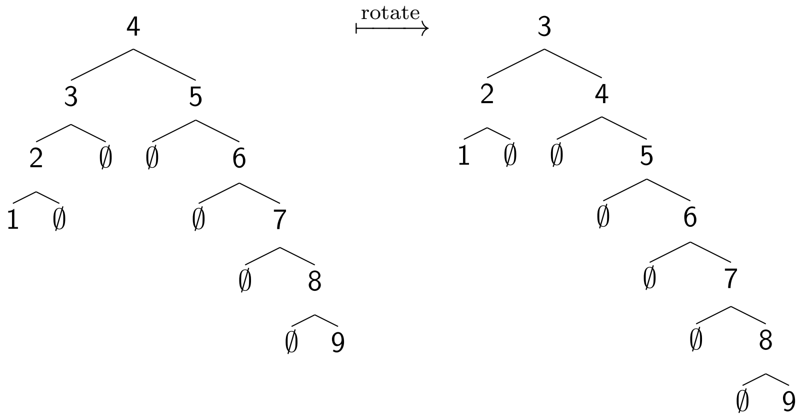
Binary Search Trees – Proof by example

- Every tree can be transformed into a "normal form" (i.e. list)



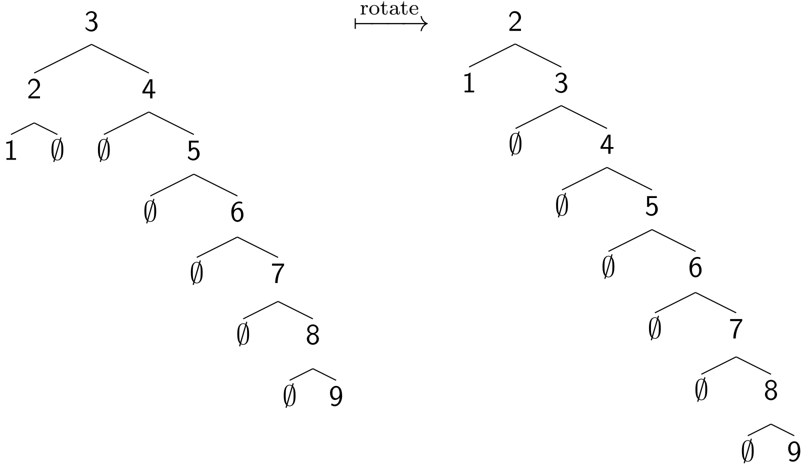
Binary Search Trees – Proof by example

- Every tree can be transformed into a "normal form" (i.e. list)



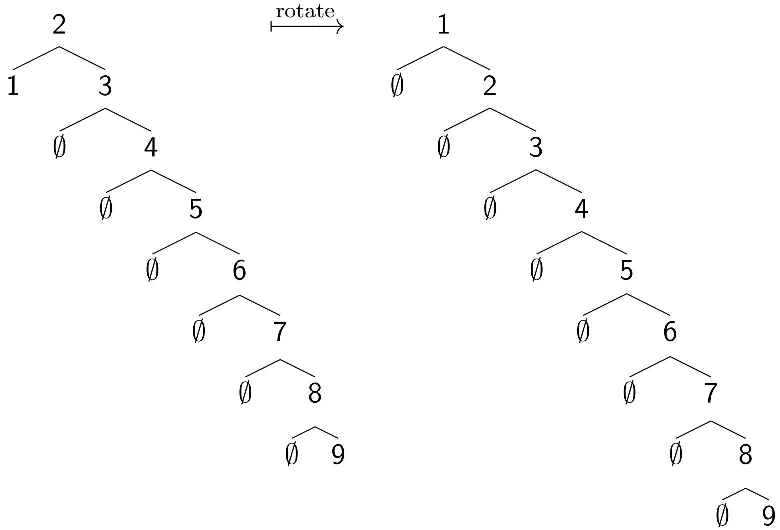
Binary Search Trees – Proof by example

- Every tree can be transformed into a "normal form" (i.e. list)



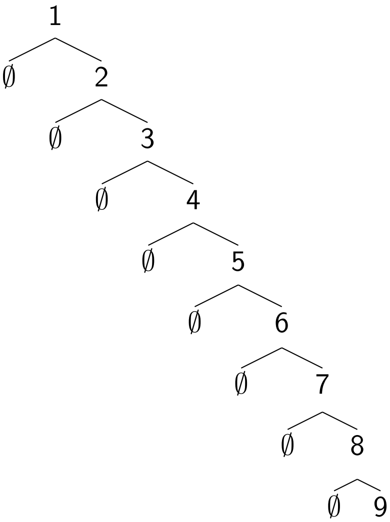
Binary Search Trees – Proof by example

- Every tree can be transformed into a “normal form” (i.e. list)



Binary Search Trees – Proof by example

- Every tree can be transformed into a “normal form” (i.e. list)



Checking BST code

- Goal: write a secondSmallest function on BSTs

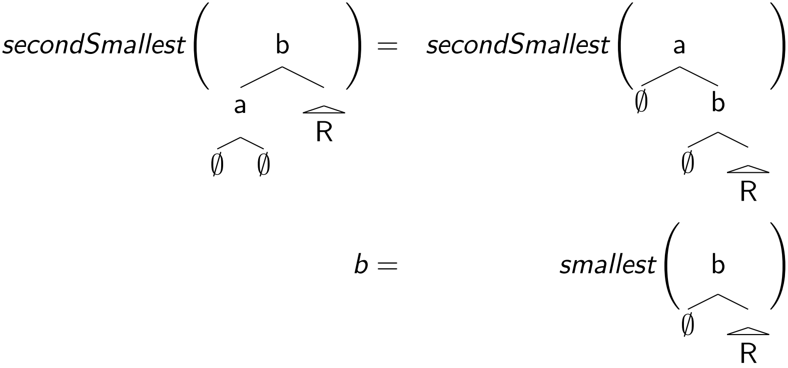
$$\text{secondSmallest}\left(\begin{array}{ccc} & b & \\ a & & \widehat{R} \\ \emptyset \quad \emptyset & & \end{array}\right) = b$$

$$\text{secondSmallest}\left(\begin{array}{ccc} & a & \\ \emptyset & & \widehat{R} \end{array}\right) = \text{smallest}(R)$$

(more cases)

Checking BST code

- One case of checking invariance under rotation:



More General Procedure

Invariance of a program $P : T \to Z$ relative to a function $f : T \to T'$

- E.g. $f : BST \to List$

Observation: The following are equivalent:

- $f(x) = f(y) \implies P(x) = P(y)$
- There exists a program $\widetilde{P} : T' \to Z$ such that $P = \widetilde{P} \circ f$

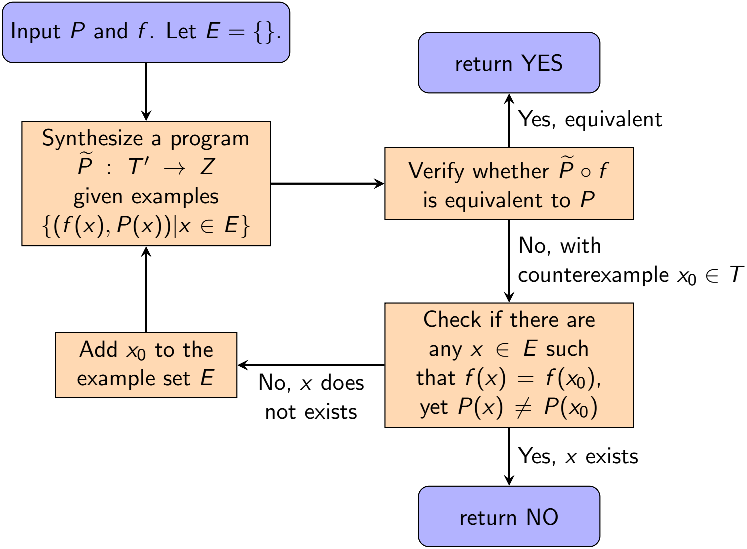
$$\begin{tikzcd}
T \arrow[rr, "P"] \arrow[dr, "f"'] & & Z \\
& T' \arrow[ur, dashed, "\widetilde{P}"'] &
\end{tikzcd}$$

More General Procedure

- Idea: Synthesize a witness to the invariance
  - A function $\widetilde{P} : T' \rightarrow Z$
- P and f provide a *full specification* of $\widetilde{P}$
- Counterexample guided inductive synthesis

$$\begin{tikzcd}
T \arrow[rr, "P"] \arrow[dr, "f"] & & Z \\
& T' \arrow[ur, dashed, "\widetilde{P}"] &
\end{tikzcd}$$

More General Procedure



More General Procedure

asdf