Verifying Robustness of Programs Under Structural Perturbations

Clay Thomas and Jacob Bond

November 29, 2017

Motivation

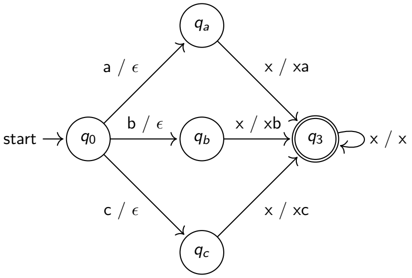
Lists – Invariance under order

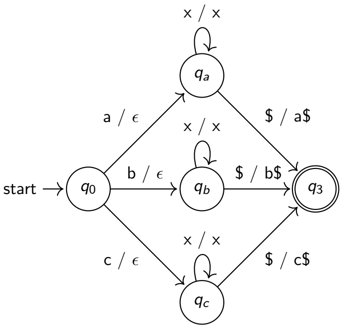
Given an array a

- 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
  - $[a[1], a[2], a[3], \ldots a[n], a[0]]$

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

Automata – Invariance under order

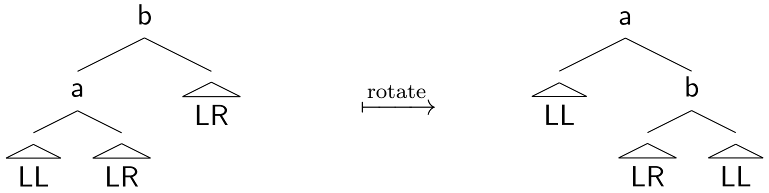


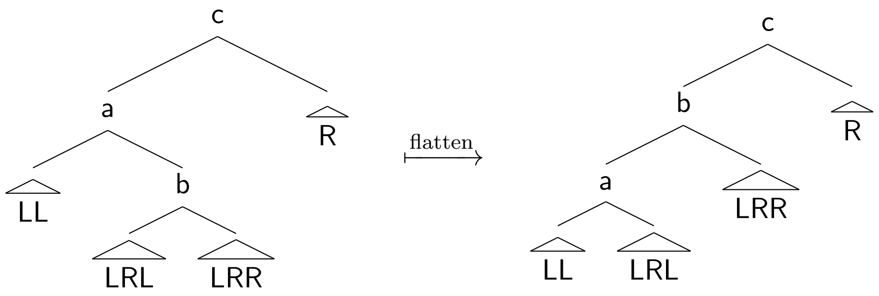


Automata – Invariance under order

Theorem: Given an automata M, we can check if M is invariant under the order of its input in time $\widetilde{O}(|\Sigma||M|)$

Binary Search Trees





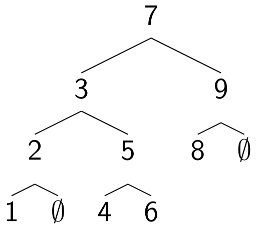
Binary Search Trees

It suffices to show

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

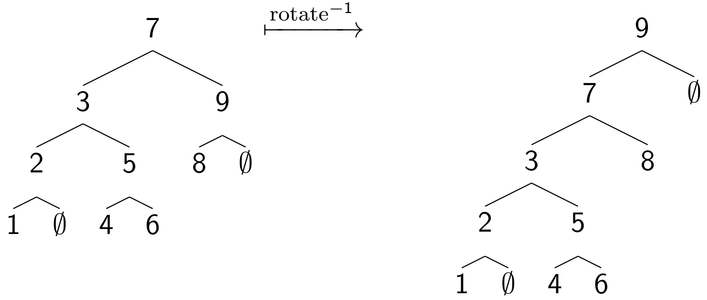
Binary Search Trees – Proof by example

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



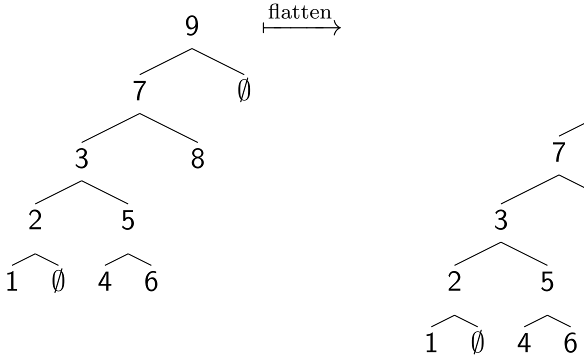
Binary Search Trees – Proof by example

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



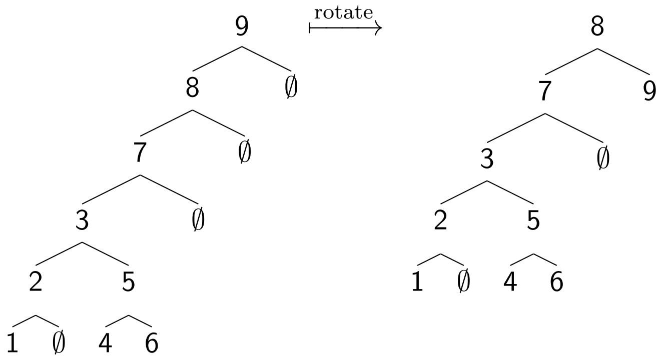
Binary Search Trees – Proof by example

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



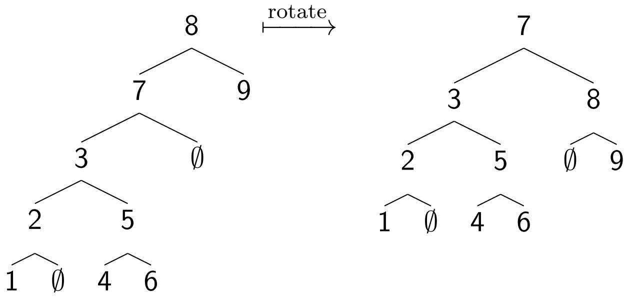
Binary Search Trees – Proof by example

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



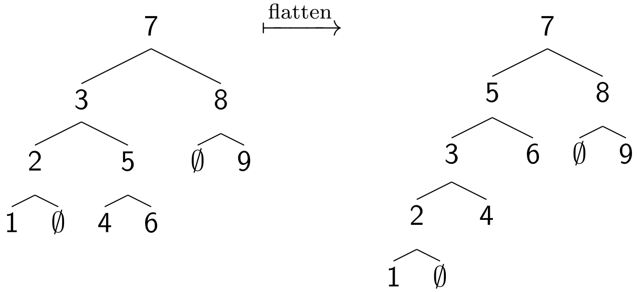
Binary Search Trees – Proof by example

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



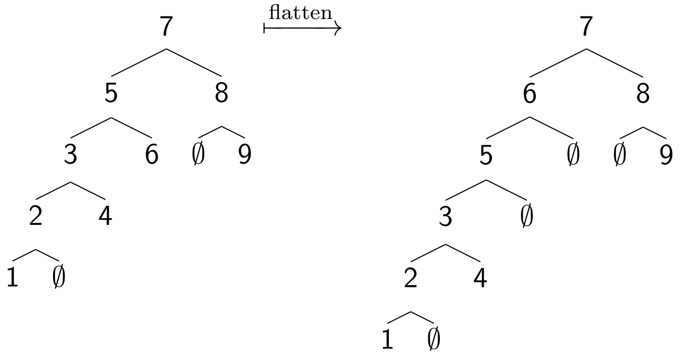
Binary Search Trees – Proof by example

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



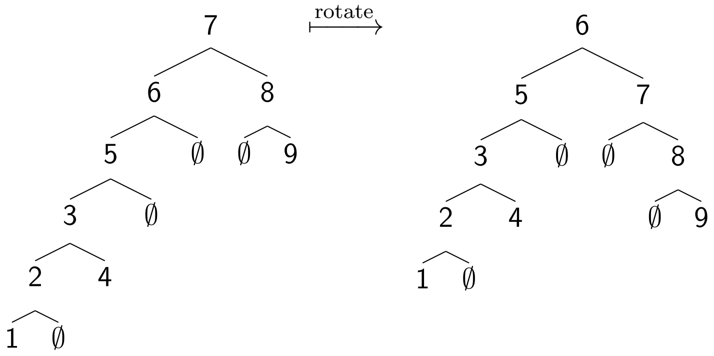
Binary Search Trees – Proof by example

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



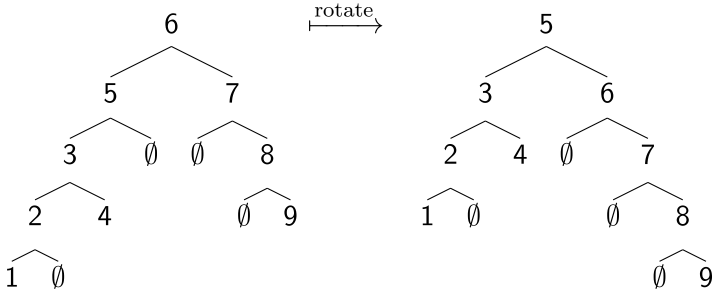
Binary Search Trees – Proof by example

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



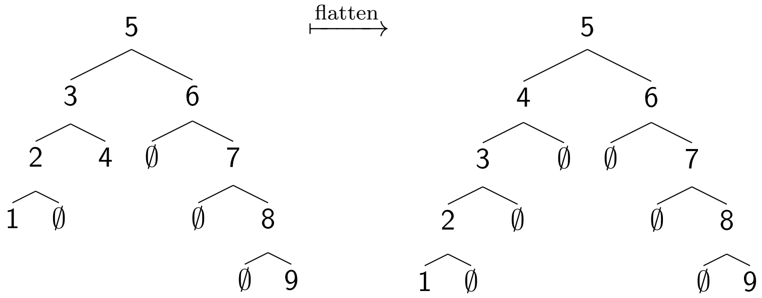
Binary Search Trees – Proof by example

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



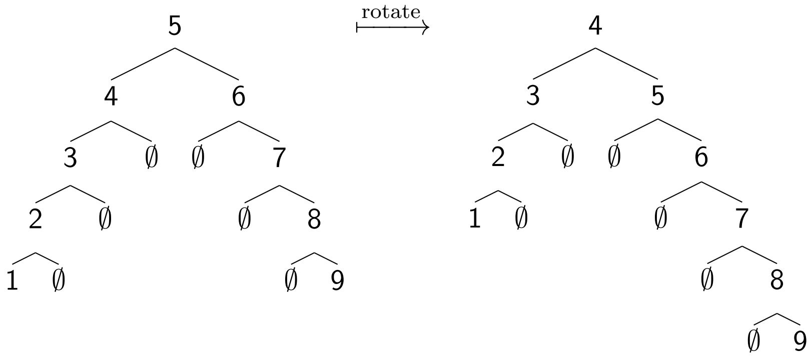
Binary Search Trees – Proof by example

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



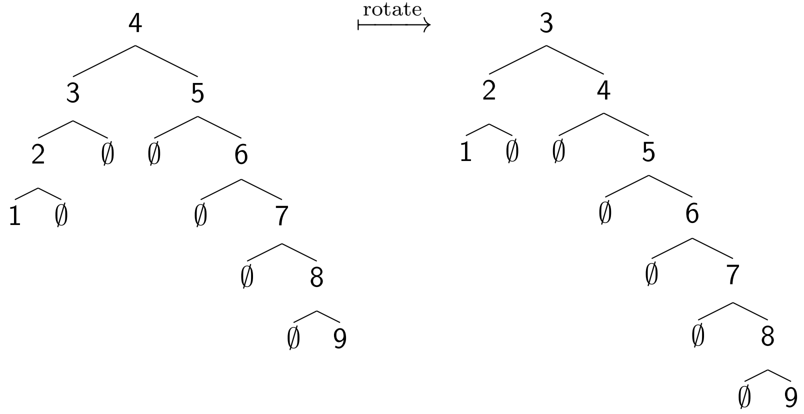
Binary Search Trees – Proof by example

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



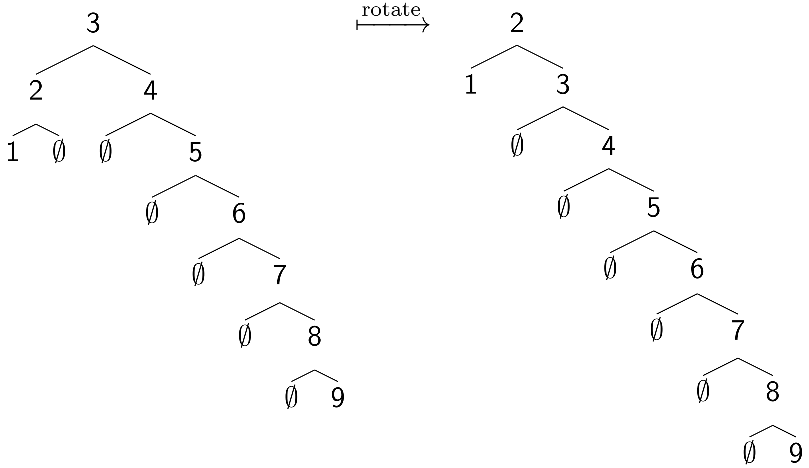
Binary Search Trees – Proof by example

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



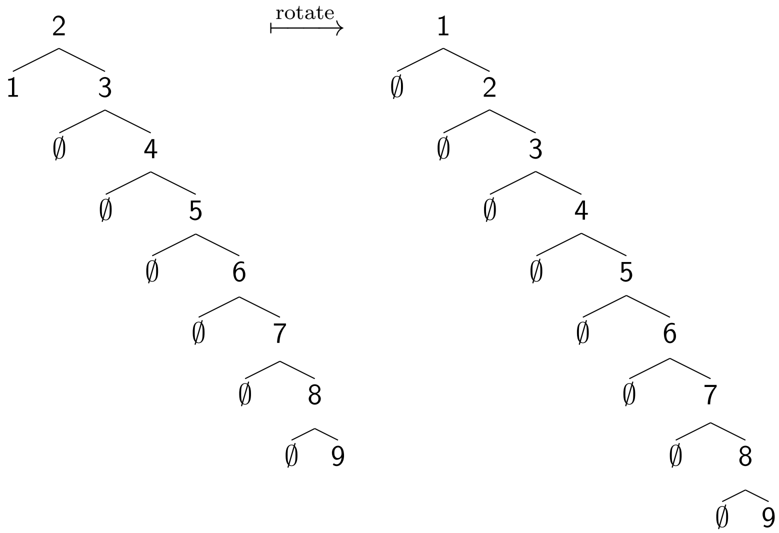
Binary Search Trees – Proof by example

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



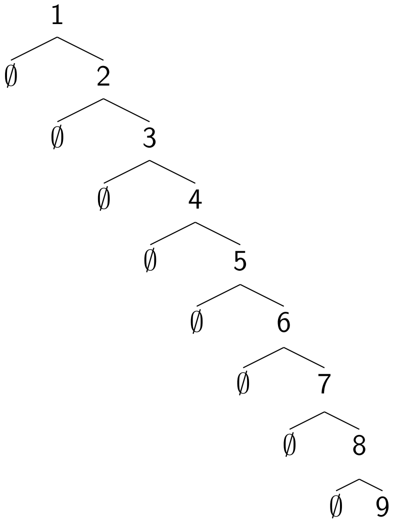
Binary Search Trees – Proof by example

- 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)



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 $P = \widetilde{P} \circ f$

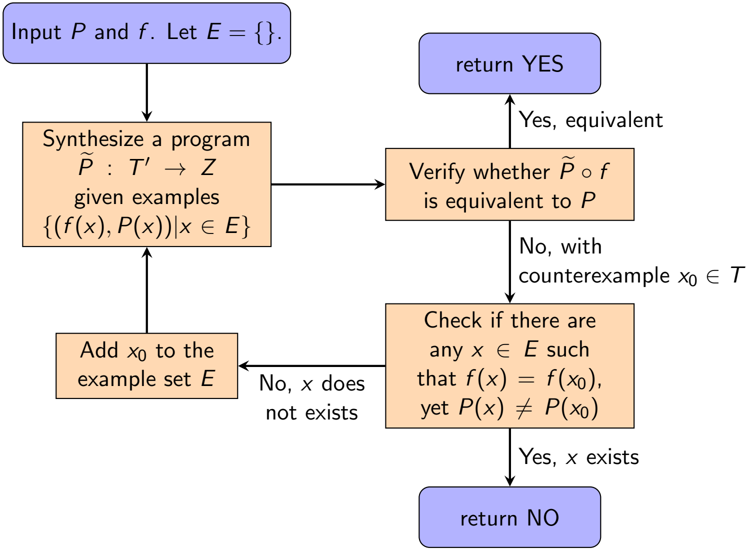
$$\begin{tikzcd}
T \arrow[rr, "P"] \arrow[dr, "f"'] & & Z \\
& T' \arrow[ur, dashed, "\widetilde{P}"'] &
\end{tikzcd}$$

More General Procedure

- Idea: Synthesize a witness to the invariance
  - A function $\widetilde{P} : T' \to Z$
- 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



Input P and f. Let $E = \{\}$.
Synthesize a program $\widetilde{P} : T' \to Z$ given examples $\{(f(x), P(x)) | x \in E\}$
Verify whether $\widetilde{P} \circ f$ is equivalent to P
Yes, equivalent
return YES
No, with counterexample $x_0 \in T$
Check if there are any $x \in E$ such that $f(x) = f(x_0)$, yet $P(x) \neq P(x_0)$
Yes, x exists
return NO
No, x does not exists
Add x_0 to the example set E

More General Procedure

asdf