


Red-Black Trees for DrACuLa

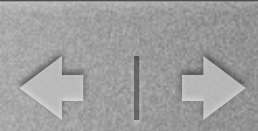
Ruben Gamboa
University of Wyoming


The Challenge

- DrACuLa (aka "Dr. ACL2") is an environment for ACL2 based on DrScheme
- It has been used in freshman-senior courses that use ACL2
- It supports a subset of the ACL2 "language"
- Rex Page asked me to update the Red-Black Tree code for DrACuLa


Red-Black Trees

- Red-Black (RB) Trees are memory ("record") data structures
- Implemented as binary search trees with some extra conditions
- Operations use the standard binary search tree ops, then perform some "rotations" to re-balance the tree

Porting to DrACuLa

- Step 0: Get DrACuLa working...
- Step 1: port my previous RB tree code
- Initially this was a major task
- Not all ACL2 functionality supported in DrACuLa


What's Missing (Then)?

- On the first day....
  - (defstructure ...)
  - (let ...) & (let* ...)
  - (defmacro ...)
  - (encapsulate nil ...)


What's Missing (Now)?

- As I worked around the missing functionality, DrACuLa was filling in the holes
  - Almost complete support for defstructure
  - let & let* are back
  - defmacro -- not so much
  - encapsulate -- still MIA


The Original Code

- Eventually, the original code was accepted by DrACuLa with what proved to be minor modifications


Adding Functionality

- The RB tree code supports insertions and lookups
- Deletes are implemented by inserting a NIL for a given key
- This works for correctness, but leaves the tree with more nodes than necessary


Deleting from RB Trees

- Step 2: Support native node deletion
- To delete from an RB tree
  - first perform ordinary binary search tree deletion
  - then rotate around the deleted tree to rebalance the tree


Deleting the Node

- Deleting a node is complicated by the fact that the node could be an interior node
  - Note: insertions always happen at leaves
- To delete an interior node, we swap it with the largest (smallest) node in the left (right) subtree
- Then perform a rotation


Work in Progress

- The deletion code is not fully verified yet
- Rebalancing the node results in a large number of program-level cases, each of which splits into thousands (of thousands) of proof-level cases


Conclusion

- DrACuLa is an effective platform supporting large/complex proof efforts
  - (encapsulate nil ...) would be nice!
- Purely Functional Data Structures (c.f. Chris Okasaki's dissertation) are a great source of problems for verification