Certifying Graph-Manipulating C Programs via Localizations within Data Structures

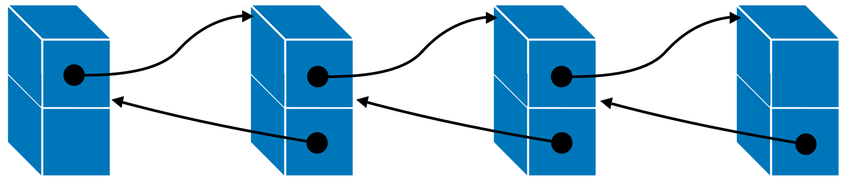
SHENGYI WANG, National University of Singapore, Singapore
QINXIANG CAO, Shanghai Jiao Tong University, China
ANSHUMAN MOHAN, National University of Singapore, Singapore
AQUINAS HOBOR, National University of Singapore, Singapore

We develop powerful and general techniques to mechanically verify realistic programs that manipulate heap-represented graphs. These graphs can exhibit well-known organization principles, such as being a directed acyclic graph or a disjoint-forest; alternatively, these graphs can be totally unstructured. The common thread LOCALIZE rule that enables modular reasoning about such programs, and show how this rule can support existential quantifiers in postconditions and smoothly handle modified program variables. We demonstrate the generality and power of our techniques by integrating them into the Verified Software Toolchain and certifying the correctness of seven graph-manipulating programs written in CompCert C, including a 400-line generational garbage collector for the CertiCoq project. While doing so, we identify two places where the semantics of C is too weak to define generational garbage collectors of the sort used in the OCaml runtime.

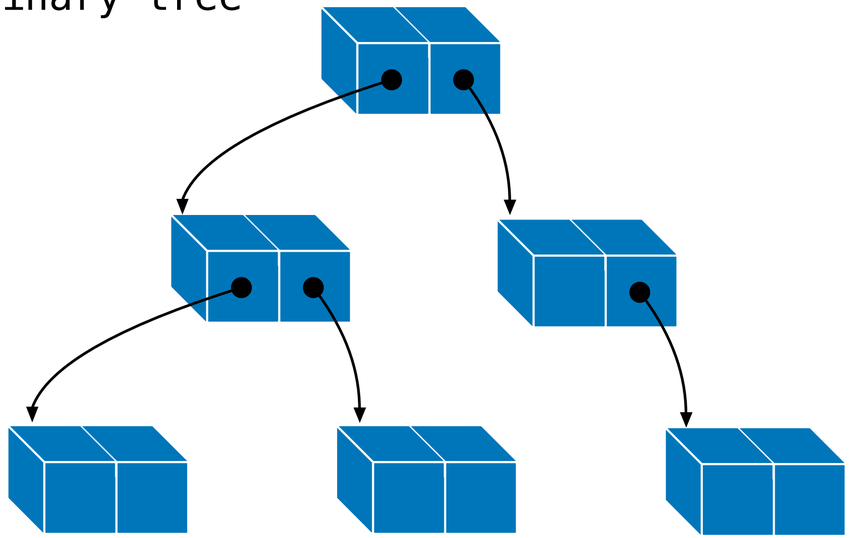
Graph-like Data Structures

singly-linked list

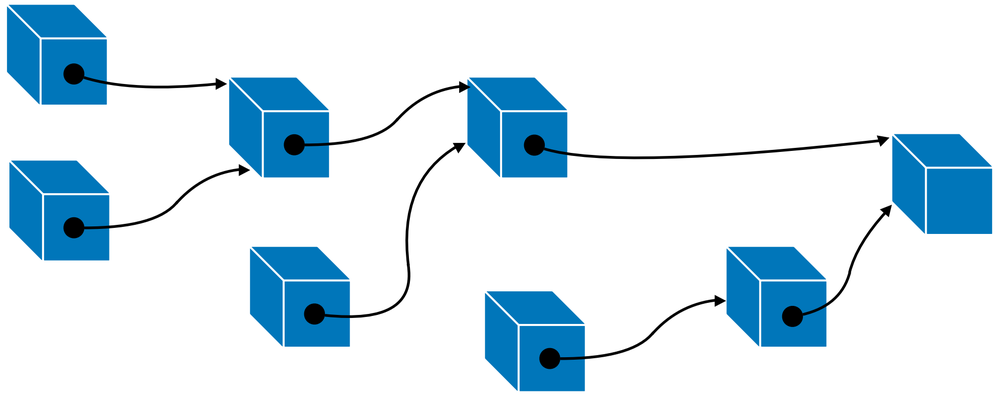
doubly-linked list



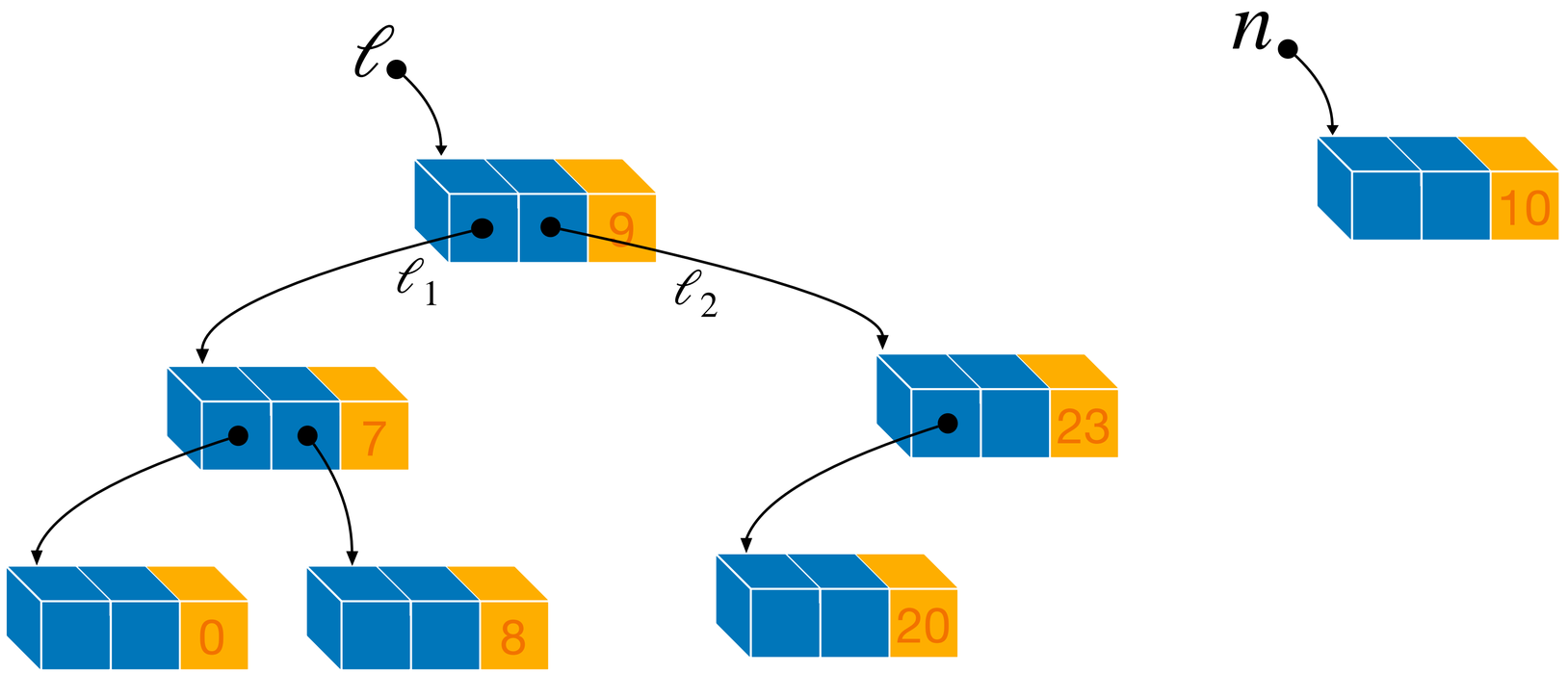
binary tree



union-find



Verifying Insert



$$\texttt{is\_Tree}\ \ell\ \texttt{t}$$

$$\Updownarrow$$

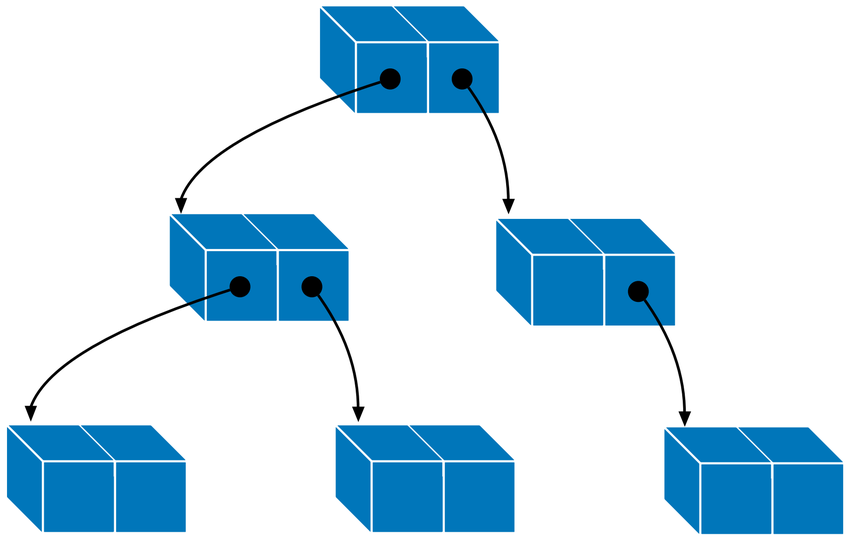
$$\ell \mapsto \ell_1, \ell_2, 9\ *\ \texttt{is\_Tree}\ \ell_1\ \texttt{t1}\ *\ \texttt{is\_Tree}\ \ell_2\ \texttt{t2}$$

Localization Rule

$$\frac{G_1 \vdash L_1 * R \qquad \{L_1\}\, p\, \{L_2\} \qquad R \vdash L_2 \mathrel{-\!\!*} G_2}{\{G_1\}\, p\, \{G_2\}}$$

Challenges

- Coming up with predicates for the heap representation
  - Specific for every graph structure?
- How to allow for local reasoning
- Dealing with graphs that have cycles



Sample of Verification work

On Paper

- Yang [2001] Graph Marking Algorithm
- Krishnaswami [2011] Union-Find

Machine-Checked

- Leino [2010] Graph Marking Algorithm
- Charguéraud [2011] Dijkstra
- Cao et al. [2018] BST (Proof Pearl)
- Guéneau et al. [2019] Cycle-Detection
- Charguéraud and Pottier [2019] Union-Find

- Hongseok Yang. 2001. Local Reasoning for Stateful Programs. Ph.D. Dissertation. University of Illinois.
- Neelakantan R. Krishnaswami. 2011. Verifying Higher-Order Imperative Programs with Higher-Order Separation Logic. Ph.D. Dissertation.
- K. Rustan M. Leino. 2010. Dafny: An Automatic Program Verifier for Functional Correctness.
- Arthur Charguéraud. 2011. Characteristic formulae for the verification of imperative programs.
- Armaël Guéneau, Jacques-Henri Jourdan, Arthur Charguéraud, and François Pottier. 2019. Formal proof and analysis of an incremental cycle detection algorithm.
- Arthur Charguéraud and François Pottier. 2019. Verifying the Correctness and Amortized Complexity of a Union-Find Implementation in Separation Logic with Time Credits
- Hobor and Villard. 2013. The ramifications of sharing in data structures.
- Cao, Wang, Hobor, and Appel. 2018. Proof Pearl: Magic Wand as Frame