Flow Interfaces

Compositional Abstractions of Concurrent Data Structures

Siddharth Krishna,

Dennis Shasha, and Thomas Wies

NYU | COURANT

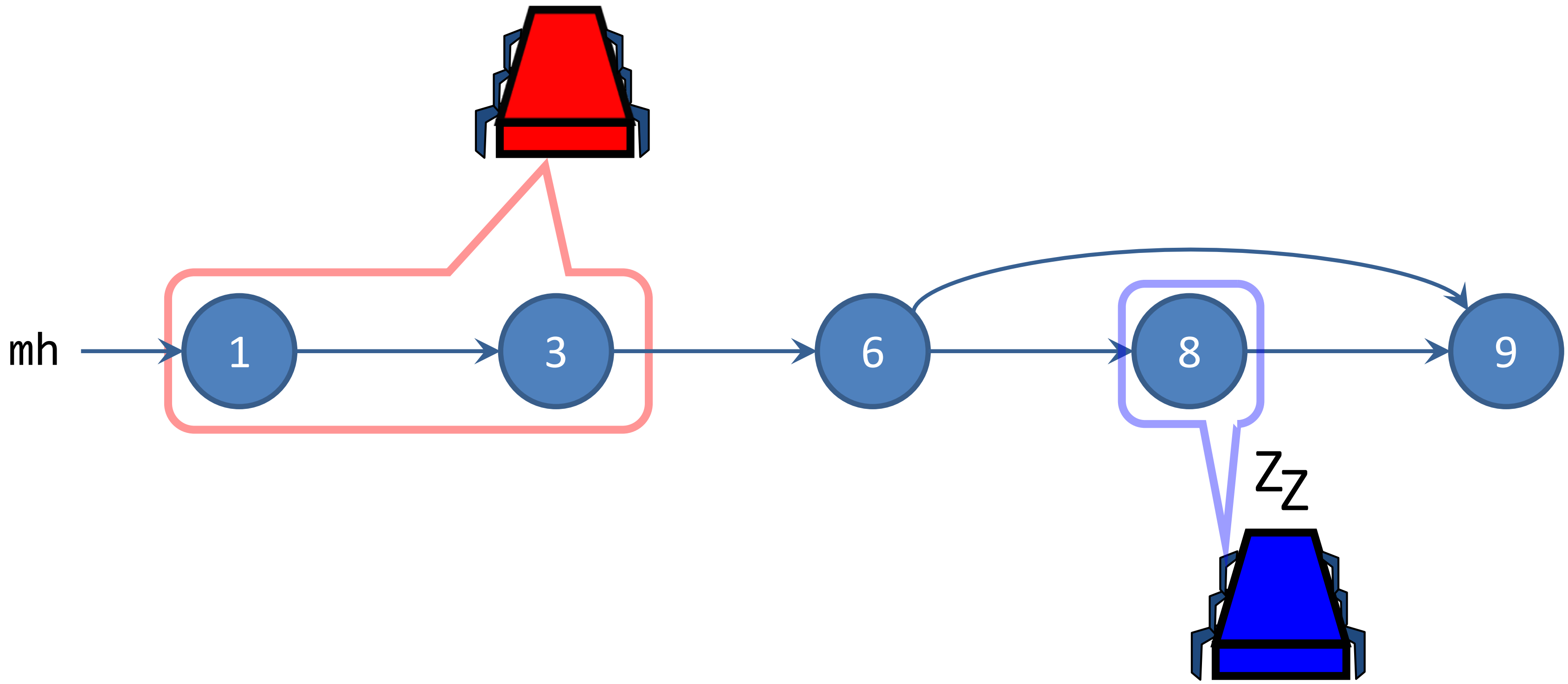
Verifying Data Structures

Separation Logic + Inductive Predicates

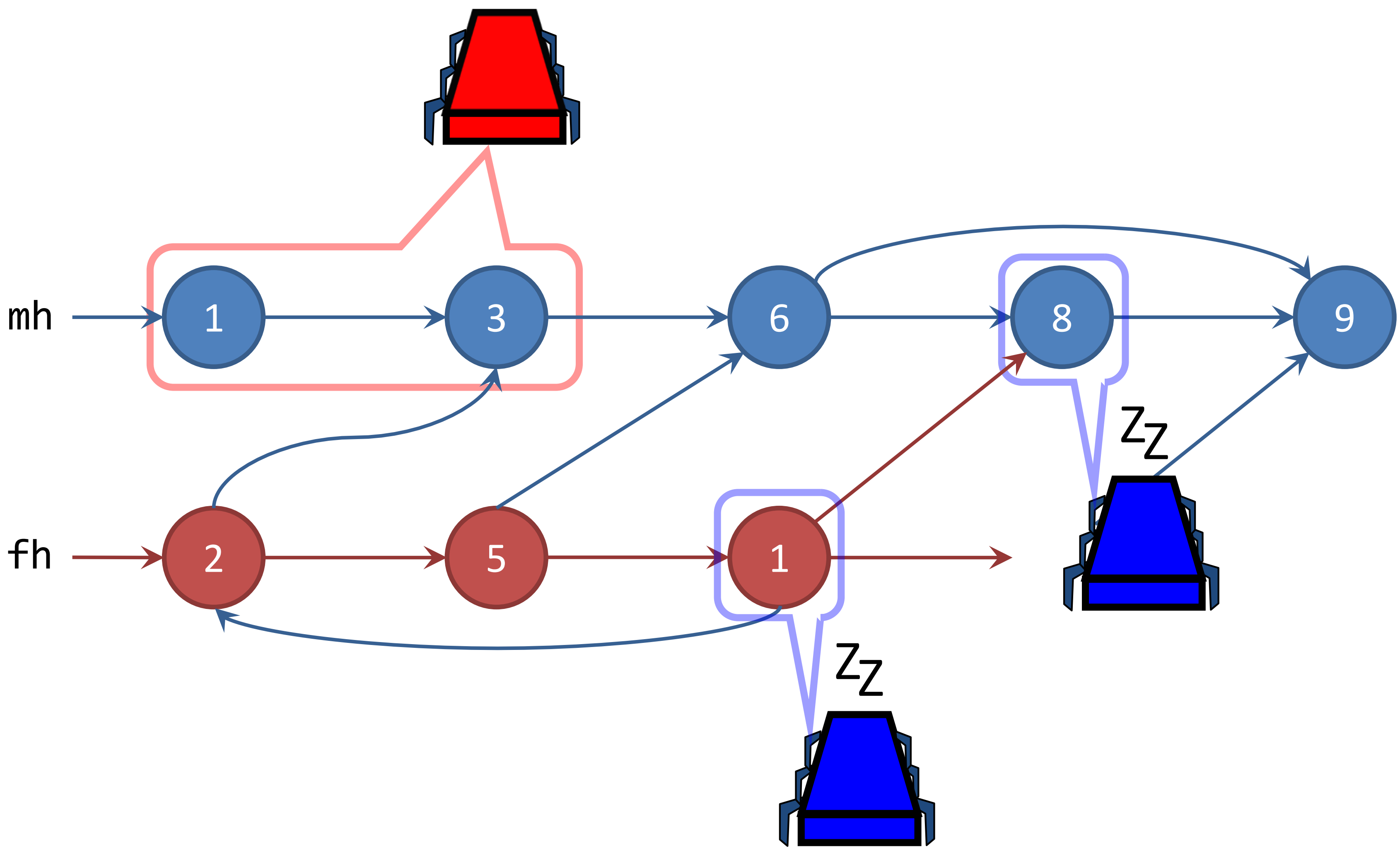
$$ls(x) \stackrel{\text{def}}{=} x = null \wedge emp \;\vee\; \exists y.\, x \mapsto y * ls(y)$$

$$\begin{aligned}
ls(x) &\rightsquigarrow \exists y. && x \mapsto y * ls(y) \\
&\rightsquigarrow \exists y, z. && x \mapsto y * y \mapsto z * ls(z) \\
&\rightsquigarrow \exists y, z, w. && x \mapsto y * y \mapsto z * z \mapsto w * ls(w) \\
&\rightsquigarrow \exists y, z, w. && x \mapsto y * y \mapsto z * z \mapsto w * w = null \wedge emp \\
&\rightsquigarrow \exists y, z, w. && x \mapsto y * y \mapsto z * z \mapsto null
\end{aligned}$$

Harris' Non-blocking List



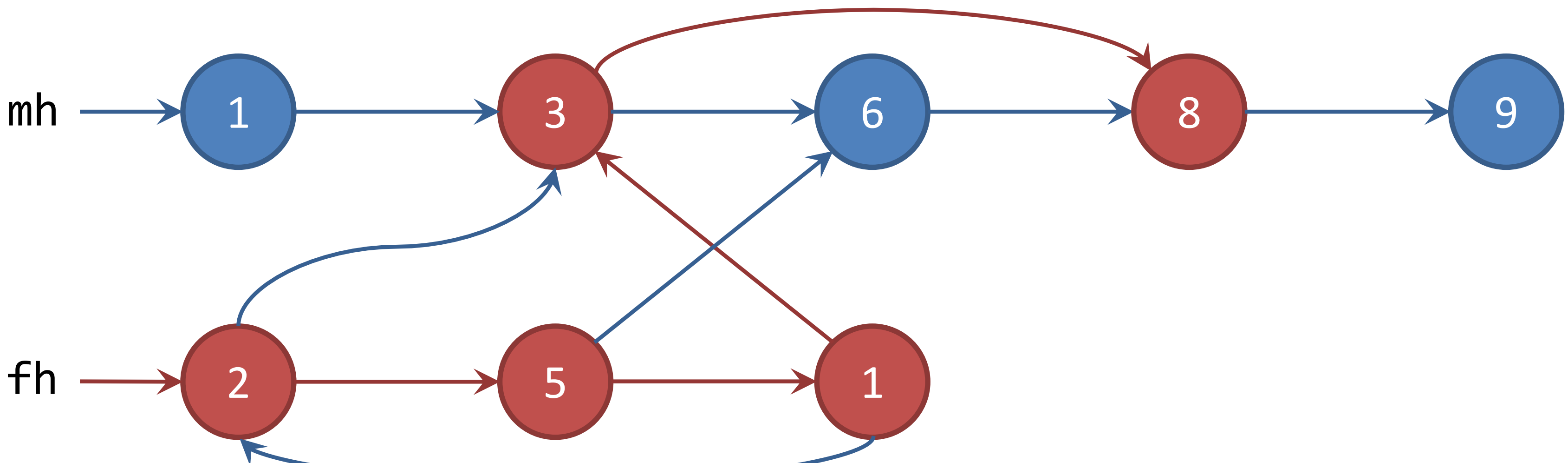
Harris' Non-blocking List



Limitations of Inductive Predicates

Traversals need to visit each node exactly once.

$$ls(mh, null) * ls(fh, null)$$

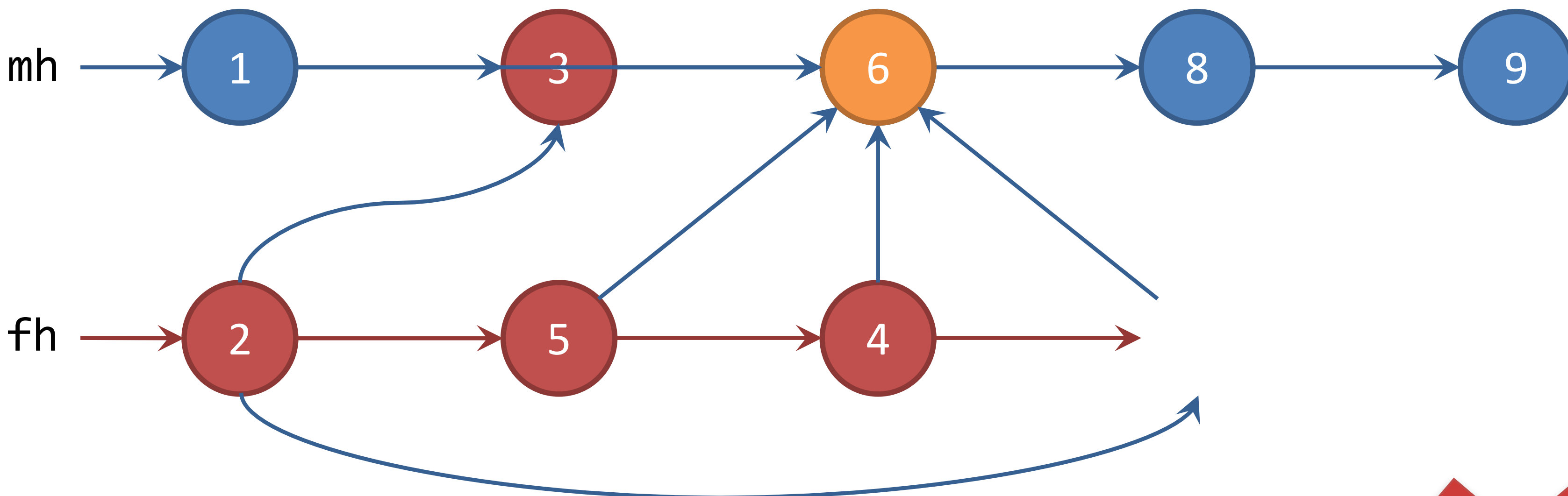


Overlays

Limitations of Inductive Predicates

Traversals need to visit each node exactly once.

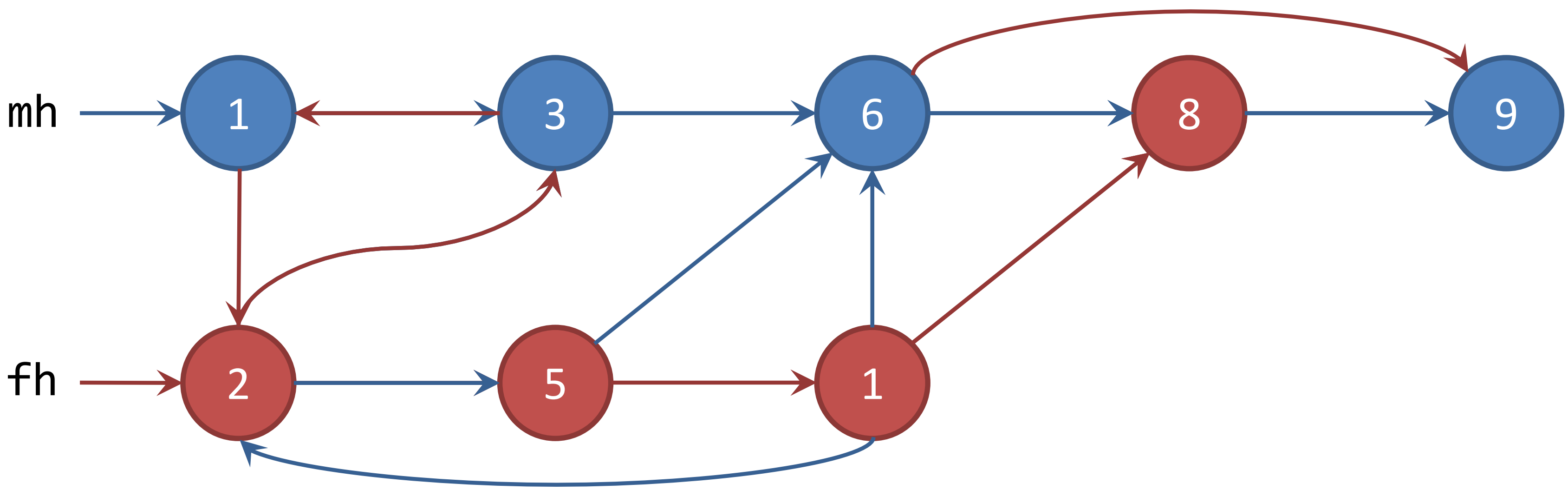
$$harris(mh, fh, null) \stackrel{\text{def}}{=} \ldots * harris(\ldots)$$



Unbounded sharing

Other Approaches

Iterated separating conjunction: $\circledast_{x \in X} \, \phi(x)$



Can’t do better than closed set of nodes

Every node reachable – memory leaks

The Problem

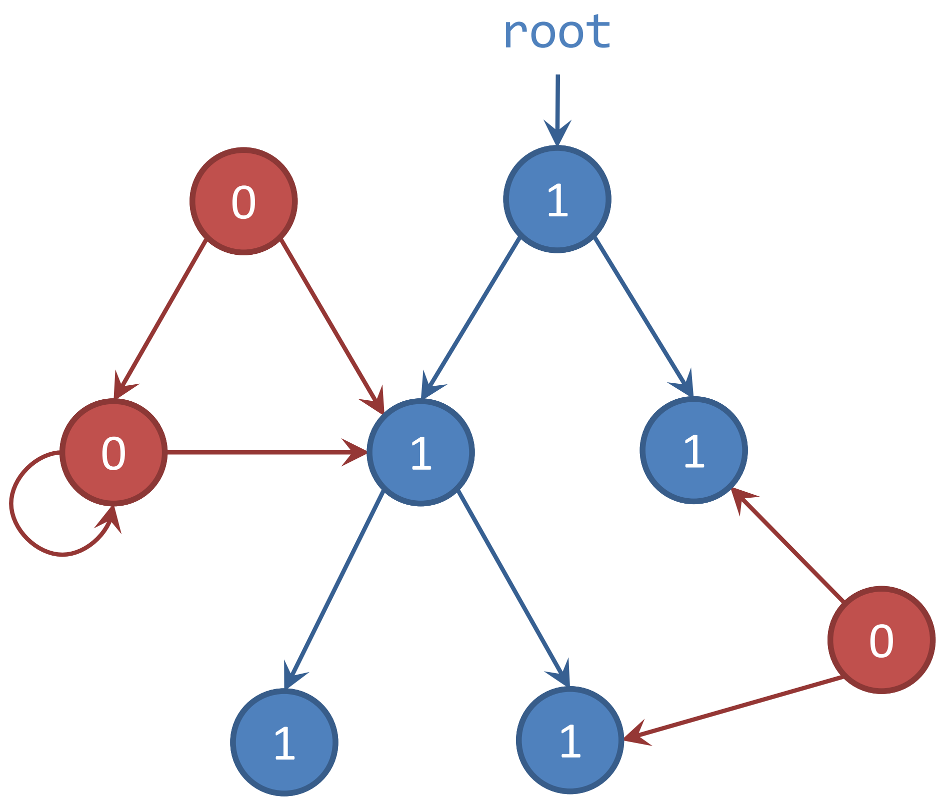
An abstraction mechanism that can handle data structures like the Harris list?

(i.e. handle overlays, sharing, arbitrary traversals & have easy reasoning)

The Idea

- Inductive predicates:
  - Pro: inductive properties
  - Con: fixed traversals – no overlays/sharing
- Iterated $*$: $\circledast_{x \in X} \phi(x)$
  - Pro: easy reasoning
  - Con: only local properties
- Best of both?
- Inductive properties $\rightarrow$ local conditions
- But allow dependence on inductive quantity: flow

Local Data Structure Invariants with Flows



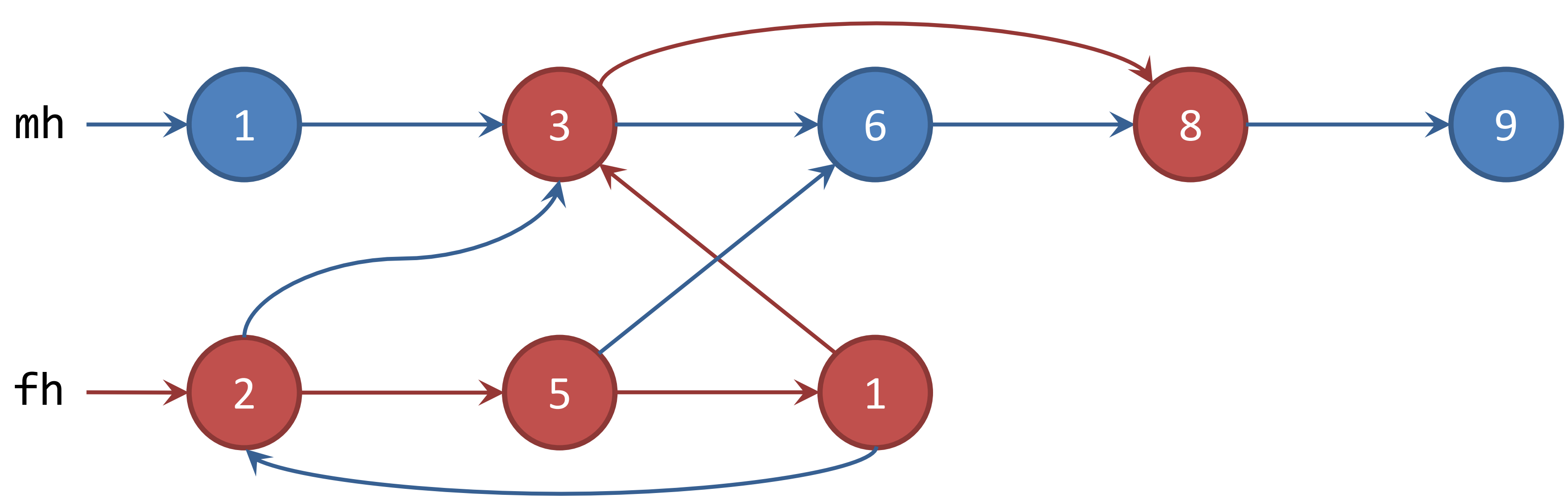
Path counting!

$$\forall n.\ pc(n) \leq 1$$

Can we express the property that `root` points to a tree as a local condition of each node in the graph?

Harris List

- Use two path-counting flows
  - One from mh and one from fh
  - Every node is on at least one of these lists
- Nodes labelled: marked/unmarked
  - All nodes in free list are marked



Separation Logic with Flows

- Graphs + Flows are a separation algebra
  - Use as semantic model
- Flow Interface: abstraction of flow graphs
  - To reason about modifications
- Logic: Gr(I)
  - A flow graph with interface I
- Expressivity:
  - Lists, trees, nested, sortedness, threaded, DAGs,…

Data-Structure-Agnostic Lemmas

- Decomposition

$$\mathsf{Gr}(I) \wedge x \in I \;\vDash\; \exists I_1, I_2.\, \mathsf{N}(x, I_1) * \mathsf{Gr}(I_2) \wedge I = I_1 \oplus I_2$$

- Step

$$I = I_1 \oplus I_2 \wedge (x, y) \in I_1^f \wedge I^f = \epsilon \;\vDash\; y \in I_2$$

- Composition

$$\mathsf{Gr}(I_1) * \mathsf{Gr}(I_2) \;\vDash\; \mathsf{Gr}(I_1 \oplus I_2)$$

Conclusion

- Flow Interfaces
  - can handle unbounded sharing and overlays
  - treat structural and data constraints uniformly
  - do not encode specific traversal strategies
  - provide data-structure-agnostic composition and decomposition rules
  - remain within general theory of separation logic
- Also: generic linearizable dictionary algorithm

Siddharth Krishna, Dennis Shasha, and Thomas Wies. Go with the Flow: Compositional Abstractions for Concurrent Data Structures. POPL 2018.