Flow Interfaces

Compositional Abstractions for Concurrent Data Structures

Siddharth Krishna, Dennis Shasha, and Thomas Wies

NYU | COURANT

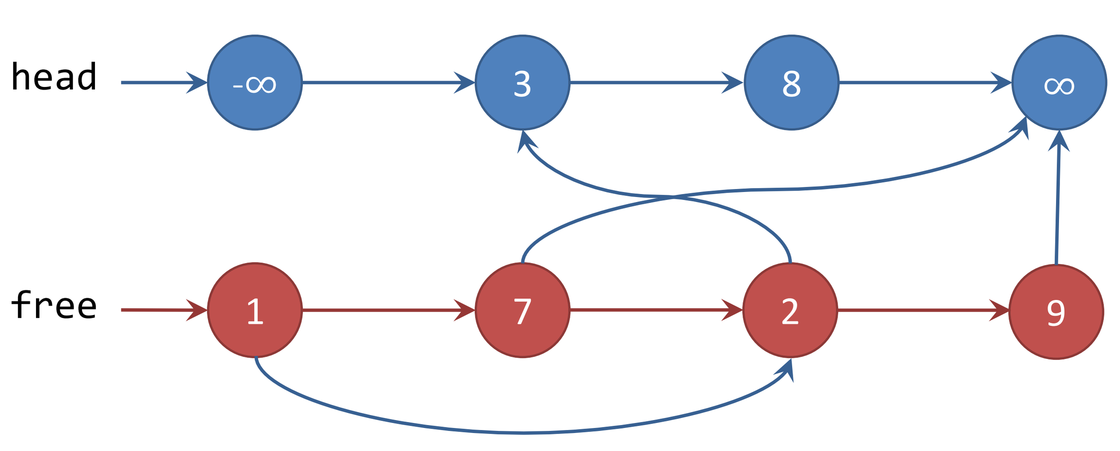
Motivation

Verifying concurrent data structures by only reasoning about the small region modified by each thread (compositional reasoning).

Challenges

- Unbounded sharing and complex overlays
- Data invariants depend on global shape

Examples: Harris' non-blocking list (below), B-link trees



Current approaches

- Separation logic (SL) based logics
- *Inductive predicates* to describe shape and data properties
- Example: list segments

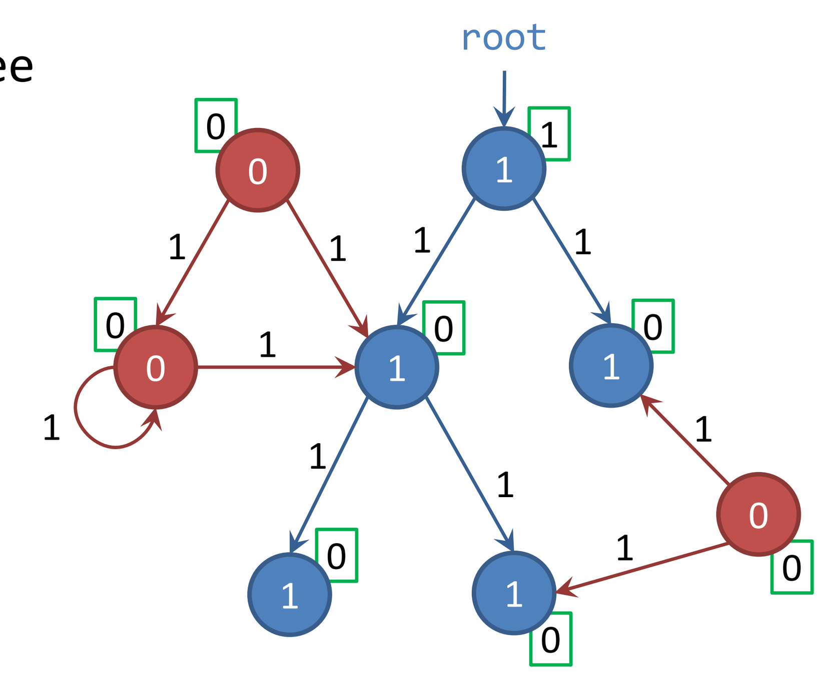
$$ls(x,y) \coloneqq (x = y \land emp) \lor (\exists z.\ x \mapsto z \ * \ ls(z,y))$$

- **Problem 1:** definition tied to traversal that visits every node exactly once
  - How do we describe Harris' list?
- **Problem 2:** predicates and lemmas are data-structure-specific
  - List composition: $ls(x,y) \ * \ ls(y,z) \Rightarrow ls(x,z)$
  - Sorted list segment with upper and lower bounds: $sls(x,y,l,u)$
  - Different composition: $sls(x,y,l,v) \ * sls(y,z,w,u) \land v \le w \Rightarrow sls(x,z,l,u)$

Flows

Key idea: encode global data invariants as local conditions on the *flow* of nodes, an inductively computed quantity.

Example specification: nodes reachable from `root` form a tree
Solution: compute number of paths from `root` to each node



Start with a *flow domain* $(D, \sqsubseteq, +, \cdot, 0, 1)$ – here use $\mathbb{N}$.

$G = (N, e)$ is a *flow graph*

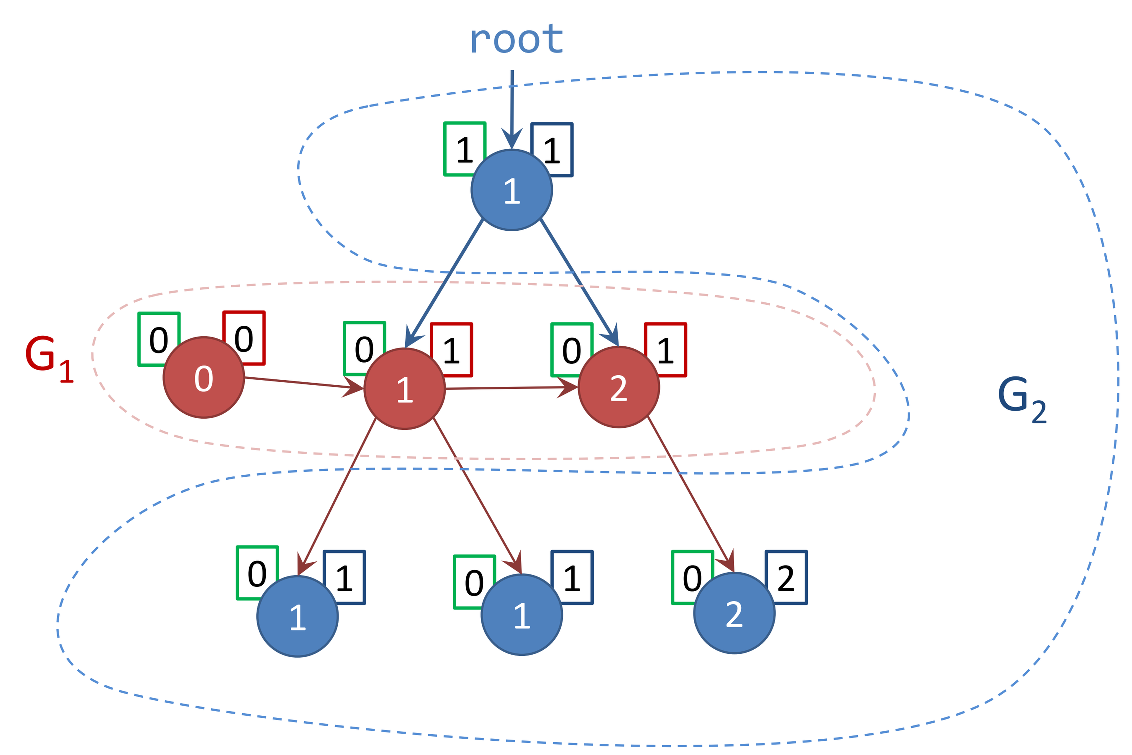
- N: finite set of nodes
- e: labels edges from D

Given an inflow $in\colon N \to D$, compute

$\mathsf{flow}(in, G) : N \to D$

$\mathsf{flow}(in, G) = \mathsf{lfp}\left(\lambda C.\, \lambda n \in N.\, in(n) + \sum_{n' \in N} C(n') \cdot e(n', n)\right)$

Example spec is now: $\forall n \in N.\ \mathsf{flow}(in, G)(n) \le 1$



Flow Interface Algebras

(in, G) is a *flow interface graph*

- G: partial flow graph with outgoing edges
- in: inflow on G

Composition and decomposition:

- Defined inductively to preserve flows
- Example: $(in, G) = (in_1, G_1) \circ (in_2, G_2)$

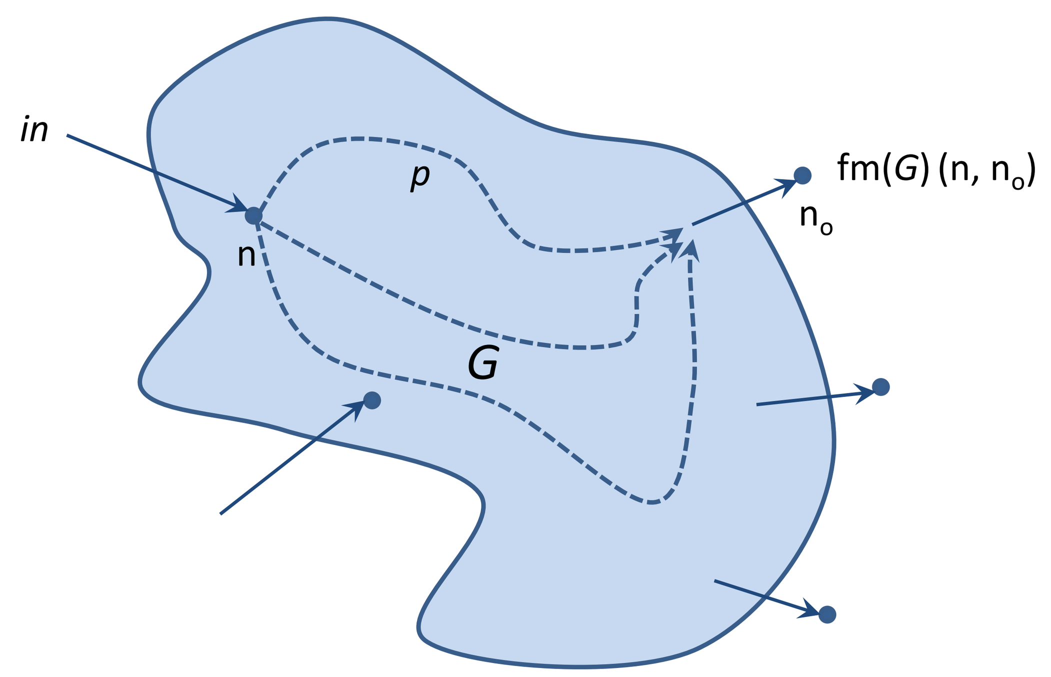
(Flow interface graphs, $\circ$) is a separation algebra
⇒ Can use as semantic model for SL

Abstractions: Flow Interfaces

- Flow map of a flow graph:

$$f \ = \ \mathsf{fm}(G)(n, n_o) = \sum_{p : n \rightsquigarrow n_o} \mathsf{pathproduct}(p)$$

- $I = (in, f)$ is a *flow interface*
- Lift composition to interfaces: $I_1 \oplus I_2$
- $[\![(in, f)]\!]_{\mathsf{good}}$ denotes all (in, G) s.t.
  - f is flow map of G
  - $\forall n \in G.\ \mathsf{good}(n, \mathsf{flow}(in, G)(n), G|_n)$ holds
- Example:
  - $\mathsf{good}(n, p, \_) = p \le 1$



Some nice properties:

- $\oplus$ is associative & commutative
- $[\![I_1]\!] \circ [\![I_2]\!] \subseteq [\![I_1 \circ I_2]\!]$

Highlights

- Separation logic based abstraction
- Handles unbounded sharing & overlays
- Local reasoning for shape and data
- Not tied to one traversal strategy
- Data-structure-agnostic composition and abstraction lemmas
- Simple correctness proofs for complex concurrent dictionary algorithms

Logic & Entailments

- Can use any concurrent SL-like logic
- To demonstrate, we use rely-guarantee separation logic (RGSep)
- We add new predicates
  - These are parametrized by the good condition

$\mathsf{Gr}(I)$	Graph region satisfying interface I
$\mathsf{N}(n, I)$	Singleton graph at n satisfying I

- Generic composition and decomposition:

$$\frac{\mathsf{Gr}(I) \land x \in I^{in}}{\mathsf{N}(x, I_1) * \mathsf{Gr}(I_2) \land I \in I_1 \oplus I_2} \quad (\textsc{Decomp})$$

$$\frac{\mathsf{Gr}(I_1) * \mathsf{Gr}(I_2) \land I \in I_1 \oplus I_2}{\mathsf{Gr}(I)} \quad (\textsc{Comp})$$

Application: Verifying Concurrent Dictionaries

We can prove memory safety and linearizability of

- Harris' non-blocking singly linked list
- B+ trees with give-up based fine grained locking

Both use same flow abstraction and key invariants for linearizability

Example: spec of B+ tree split method:

$$\left\{\begin{array}{l}\boxed{(\mathsf{N}(p, I_p) * \mathsf{N}(c, I_c)) \twoheadrightarrow\!\!* \ \mathsf{Gr}(I)} \land I^{in} = \{r \mapsto (\mathsf{KS}, 1)\}.\mathbf{0} \\ \land I^f = \epsilon \land I^f_p(p, c) \ne (\emptyset, 0) \land I^\alpha_p = (C_p, t) \land I^\alpha_c = (C_c, t)\end{array}\right\}$$

```
split(c, p);
```

$$\left\{\begin{array}{l}\boxed{(\mathsf{N}(p, I'_p) * \mathsf{N}(c, I'_c) * \mathsf{N}(n, I_n)) \twoheadrightarrow\!\!* \ \mathsf{Gr}(I)} \land I^{in} = \{r \mapsto (\mathsf{KS}, 1)\}.\mathbf{0} \\ \land I^f = \epsilon \land I'^\alpha_p = (C_p, t) \land I'^\alpha_c = (C'_c, t) \land I'^\alpha_n = (C_n, t) \land C_c = C'_c \cup C_n\end{array}\right\}$$