### Flow Interfaces

**Compositional Abstractions for Concurrent Data Structures**

**Siddharth Krishna, Dennis Shasha, and Thomas Wies**

---

**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
  
  \[ ls(x,y) = (x = y \land \text{emp}) \lor (\exists z. x \leftrightarrow z \land ls(z,y)) \]

- 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:
    
    \[ ls(x,y) \land ls(y,z) \Rightarrow ls(x,z) \]
  - Sorted list segment with upper and lower bounds:
    
    \[ ls(x,y,z,u) \land \exists v. x \leftrightarrow v \land ls(z,v,u) \]
  - Different composition:
    
    \[ ls(x,y,z,w,u) \land v \leq w \Rightarrow ls(z,x,z,u) \]

---

**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 \(N\).

\[ G = (N, e) \text{ is a flow graph} \]
- \(N\): finite set of nodes
- \(e\): labels edges from \(D\)

Given an inflow \(in: N \rightarrow D\), compute

\[ \text{flow}(in,G) : N \rightarrow D \]

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

**Example spec is now:** \(\forall n \in N. \text{flow}(in,G)(n) \leq 1\)

---

**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: \((in,G) = (in_2,G_2) \circ (in_1,G_1)\)

(Flow interface graphs, \(\circ\) is a separation algebra

\[ \Rightarrow \text{Can use as semantic model for SL} \]

---

**Application: Verifying Concurrent Dictionaries**

We can prove memory safety and linearity 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 linearity

**Example:** spec of B+ tree split method:

\[
\begin{align*}
\left[ \text{in}(\text{p},\text{G}) \land \text{in}(\text{c},\text{L}) \right] \rightarrow \text{Gr}(G) \land \text{l}^0 = \{ r \Rightarrow (\text{KS},1) \} \}

\land \begin{aligned}
\text{if} &\ = e \land \text{l}^0_p \land (\text{b} \land \text{c}(\text{p},\text{c}) \land (\text{p} \land \text{c}(\text{c},\text{c})) \land \text{Gr}(\text{g})) \land \text{l}^0_m \land (\text{r} \Rightarrow (\text{KS},1)) \}
\end{aligned}
\end{align*}
\]

---

**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 parameterized by the good condition
    
    \[ \text{Gr}(f) \]
    
    Graph region satisfying interface \(f\)
  - Generic composition and decomposition:
    
    \[ \text{Gr}(f) \land x \in 1^m \]
    
    \[ \text{N}(x,I) \land \text{Gr}(I) \land I \in I_1 \oplus I_2 \] (DECOMP)
    
    \[ \text{Gr}(I_1) \land \text{Gr}(I_2) \land I \in I_1 \oplus I_2 \]
    
    \[ \text{Gr}(f) \] (COMP)

---

**Some nice properties:**
- \(\otimes\) is associative & commutative
- \(l_1 \cdot l_2 \land l_1 \leq l_2\)

---

**Compositional Flow Interfaces**

\[ f = \text{fm}(G)(n, n_o) = \sum_{p \in N \land n_o} \text{pathprod}(p) \]

\[ l = (in,f) \text{ is a flow interface} \]

Lift composition to interfaces: \(l_1 \oplus l_2\)

\[ \{ (in,f) \}_\text{good} \text{ denotes all } (in,G) \text{ s.t.} \]

\[ f \text{ is flow map of } G \]

\[ \forall n \in G, \text{ good}(n, \text{flow}(in,G)(n), G|_n) \text{ holds} \]

**Example:**
- \(\text{good}(n, p, \_ ) = p \leq 1\)