

# Regional Logic for Local Reasoning about Global Invariants

Anindya Banerjee<sup>1,\*</sup>, David A. Naumann<sup>2,\*\*</sup>, and Stan Rosenberg<sup>2,\*\*</sup>

<sup>1</sup> Kansas State University, Manhattan KS 66506 USA and  
Microsoft Research, Redmond WA 98052 USA

<sup>2</sup> Stevens Institute of Technology, Hoboken NJ 07030 USA

**Abstract.** Shared mutable objects pose grave challenges in reasoning, especially for data abstraction and modularity. This paper presents a novel logic for error-avoiding partial correctness of programs featuring shared mutable objects. Using a first order assertion language, the logic provides heap-local reasoning about mutation and separation, via ghost fields and variables of type ‘region’ (finite sets of object references). A new form of modifies clause specifies write, read, and allocation effects using region expressions; this supports effect masking and a frame rule that allows a command to read state on which the framed predicate depends. Soundness is proved using a standard program semantics. The logic facilitates heap-local reasoning about object invariants: disciplines such as ownership are expressible but not hard-wired in the logic.

## 1 Introduction

The potential for interference between supposedly independent program phrases or components due to shared mutable objects is the bane of formal reasoning and static analysis of software. This paper charts new territory, combining two simple and well known ideas —regions and ghost state— in a new way. We formulate a logic that needs only classical, first-order assertions, though inductively defined predicates are compatible. The key novelty is “modifies” specifications expressed in terms of state-dependent region expressions. Together with judicious static analysis of the “footprints” of formulas, this makes it possible to achieve the kinds of modularity associated with ownership methodologies and separation logic, in a flexible way that is compatible with widely used specification languages and tools.

Various notions of regions have been used in static analysis to abstract sets of objects of interest [28]. Separation logic [23] owes its success in specifying and verifying pointer algorithms at least in part to its ability to manifest the “footprint” or region of heap relevant to a particular predicate (and thereby the footprint of a command). At a coarser level, ownership systems and separation logic ideas have been critical to advances in data abstraction [11,2], especially for object invariants [18,5].

In this paper, instead of abstracting from regions and expressing separation via logical connectives or ownership types, we make regions explicit in a way similar to work

---

\* Partially supported by US NSF awards CNS-0627748, ITR-0326577.

\*\* Partially supported by US NSF awards CNS-0627338, CRI-0708330, CCF-0429894.

of Amtoft et al [1]. Most importantly, we follow Kassios [14] in using regions to directly represent footprints. We augment a Java-like language with type **rgn** ranging over finite sets of (allocated) references. Following Kassios, we instrument programs with assignments to ghost variables and fields, so assertions can refer explicitly to regions. Whereas Kassios works in the setting of a relational refinement theory and higher order logic, we develop a Hoare logic using first class regions in the “modifies” clause, often the most useful part of a program specification. Asserting the disjointness of regions helps delimit effects and facilitate heap-local reasoning.

It is no surprise that it is possible to reason in terms close to the semantic model [8]. If one’s aim is to prove functional correctness of, say, a garbage collector then at the very least, the specification involves reachability, inductive definitions, quantification over paths, etc. But to specify and prove weaker properties, e.g., that an application program does not stray beyond its intended resources, what we achieve is promising. Without the need for inductive predicates or quantification over predicates to hide all but their footprint, we reason directly in terms of footprints. In particular we get “frame rules” that account for modular reasoning about representation invariants.

Notions like ownership [11] support encapsulation of state on which a single object’s invariant depends. A precursor to our work is the use of ghost state to encode ownership [16,22] in a way that allows transfer of objects between clients and abstractions (as in low level memory management and higher level OO design patterns like connection pools and layered I/O abstractions). Unlike ownership type systems or programming disciplines, and unlike static analyses using regions, we avoid commitment to a fixed use of regions. On the contrary, regions as ghost state can encode such disciplines but can also combine them in uniform or ad hoc ways.

A benefit of treating regions as ghost state is that it can be done using first-order specification languages based on classical logic with modest use of set theory. Thus it fits with mostly-automated tools based on verification condition generation and it fits with conventional means of program structuring such as scope-based encapsulation. In this foundational study we expose the issues and formalize the ideas in terms of a simple object-based language and Hoare-style proof system which we prove sound using a standard program semantics. There is a major difficulty: “modifies” specifications using region expressions dependent on mutable state are susceptible to a kind of interference: The effect of a command can alter the meaning of the effect specification of another command! This issue has appeared before, in Kassios’ dissertation and in the work of Leino and Nelson [17]; we explicitly focus on modifies specifications and offer a novel and flexible solution.

Our first contribution is the logic: its rules and subsidiary judgements together with proof of soundness. Various subtleties made it difficult to correctly design the details of our logic, but we find most of the rules and soundness proofs to be elegant.

Our second contribution is to show how the logic serves as a basis for encapsulating object invariants and invariants for clusters of objects (peers, friends and beyond). Remarkably, our approach can be formalized by a second order frame rule like that of separation logic. Soundness of the second order frame rule in separation logic is challenging [23,7]. Our version is an admissible rule, but the technical result is the subject of another paper [21]. In this paper, we propose an approach to developing sound