

# Boogie Meets Regions: A Verification Experience Report

Anindya Banerjee<sup>1,\*</sup>, Mike Barnett<sup>2</sup>, and David A. Naumann<sup>3,\*\*</sup>

<sup>1</sup> Kansas State University, Manhattan KS 66506, USA

<sup>2</sup> Microsoft Research, Redmond WA 98052, USA

<sup>3</sup> Stevens Institute of Technology, Hoboken NJ 07030, USA

**Abstract.** We use region logic specifications to verify several programs exhibiting the classic hard problem for object-oriented systems: the framing of heap updates. We use BoogiePL and its associated SMT solver, Z3, to prove both implementations and client code.

## 1 Introduction

Many programs use dynamically allocated mutable storage, which poses challenges for encapsulation and exacerbates the frame problem: how to specify that “everything else is unchanged” for mutable state not directly named by program variables? Several lines of work approach this problem in terms of *footprints*, i.e., the sets of locations that are written or read by a phrase of a program or specification.

- In the Boogie methodology [5], “owned state” is represented by a mutable ghost field that points to an object’s owner if it has one. An *effect specification* (“modifies” clause) that licenses update of an object  $o$  implicitly licenses update of the objects transitively owned by  $o$ .
- In Separation Logic [21], if  $P$  is the precondition of a procedure then the procedure may be viewed as owning the state on which  $P$  depends. It has license to modify that part of the heap but no other.
- Kassios [17] shows how to manipulate footprints explicitly, in ghost variables, rather than implicitly (in formulas) or indirectly (via transitive ownership).

A key point is that if the read footprint of a formula or pure expression is disjoint from the write footprint of a command, then the command preserves the value of the formula or expression.

Kassios works in a higher order logic, which can directly express that a formula depends on certain locations. He shows how reasoning idioms developed in Boogie or separation logic can be elegantly and effectively articulated using explicit footprints. His approach does require a single discipline to be imposed on all parts of all programs. He develops a relational refinement calculus, following Hehner [15].

---

\* Partially supported by US NSF awards CNS-0627748 and ITR-0326577 and by a sabbatical visit at Microsoft Research, Redmond.

\*\* Support from NSF (CNS-0627338, CRI-0708330, CCF-0429894) and Microsoft Research.

Smans et al [24] explore Kassios' approach in terms of conventional sequential specifications with distinct first order precondition, postcondition, and effect. They focus on framing of pure methods used in specifications for data abstraction. They report on encouraging case studies using a prototype verifier based on an SMT solver.

Banerjee et al [4] also explore the approach using first-order specifications, focusing on foundational justification in terms of a Hoare logic that uses footprint expressions in the effects clause. Their Region Logic features a frame rule inspired by that of separation logic (which in turn adapts Hoare's rule of Invariance) to encompass interference via mutable heap objects. In region logic, the frame rule involves a static analysis for read effects of formulas. The effect  $\mathbf{rd} \ G.f$  expresses that field  $f$  of some objects in  $G$  may be read. A region expression  $G$  denotes a set of non-null object references and footprints are of the form  $G.f$ . This treatment caters for notations like  $G.f \subseteq G'$  which says that the  $f$ -image of  $G$  is a subset of  $G'$ .<sup>1</sup> Region logic features the use of region fields and variables to encode dynamic frames whereas Smans *et al.* use pure methods. Banerjee *et al.* claim that: "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..."

This paper reports on some case studies to investigate Banerjee *et al.*'s claim. We demonstrate the utility of using region specifications for

- Local reasoning about data structures (Sect. 4),
- Encapsulation, even in the presence of callbacks (Sect. 5),
- Layered abstractions, using the examples from Smans et al. [24] but without the need for conditional effects (Sect. 6).

In addition, our experience supports their suggestion that the region logic provides a neutral framework in which disciplines such as ownership can be, but need not be imposed system-wide.

In ongoing work we are investigating decision procedures for region logic primitives such as  $G.f \subseteq G'$ . Here, we use the Z3 solver [12] which does not have a decision procedure for set theory. This poses another research question: Is a first order axiomatization of set theory effective for automating verification of programs and specifications that encode footprints in ghost state? Our answer is yes (Sect. 9) and no (Sect. 7).

Rather than implement a verification condition generator from scratch for our assertion language, we use the BoogiePL intermediate language advocated at VSTTE'05 by Barnett et al [6]. The BoogiePL tool [13] generates verification conditions suited for SMT solvers such as Simplify [14], the CVC family [8], and Z3 [12]. These integrate multiple decision procedures with heuristic instantiation of quantifiers driven by pragmas called *triggers*. They have been used with some success in verifiers like ESC/Java and Spec#, where user interaction is limited to the insertion of assert and assume statements and loop invariants. Our experiments consist entirely of manually written encodings in BoogiePL of the example programs and specifications, with Z3 as solver. We are therefore in a position to provide triggers on an ad hoc basis.

---

<sup>1</sup> These features are useful for reasoning about simulations, as in the context of information flow [1,2].