

# Local Reasoning about Programs that Alter Data Structures

Peter O'Hearn<sup>1</sup>, John Reynolds<sup>2</sup>, and Hongseok Yang<sup>3</sup>

<sup>1</sup> Queen Mary, University of London

<sup>2</sup> Carnegie Mellon University

<sup>3</sup> University of Birmingham and University of Illinois at Urbana-Champaign

**Abstract.** We describe an extension of Hoare's logic for reasoning about programs that alter data structures. We consider a low-level storage model based on a heap with associated lookup, update, allocation and deallocation operations, and unrestricted address arithmetic. The assertion language is based on a possible worlds model of the logic of bunched implications, and includes spatial conjunction and implication connectives alongside those of classical logic. Heap operations are axiomatized using what we call the "small axioms", each of which mentions only those cells accessed by a particular command. Through these and a number of examples we show that the formalism supports local reasoning: A specification and proof can concentrate on only those cells in memory that a program accesses.

This paper builds on earlier work by Burstall, Reynolds, Ishtiaq and O'Hearn on reasoning about data structures.

## 1 Introduction

Pointers have been a persistent trouble area in program proving. The main difficulty is not one of finding an in-principle adequate axiomatization of pointer operations; rather there is a mismatch between simple intuitions about the way that pointer operations work and the complexity of their axiomatic treatments. For example, pointer assignment is operationally simple, but when there is aliasing, arising from several pointers to a given cell, then an alteration to that cell may affect the values of many syntactically unrelated expressions. (See [20, 2, 4, 6] for discussion and references to the literature on reasoning about pointers.)

We suggest that the source of this mismatch is the global view of state taken in most formalisms for reasoning about pointers. In contrast, programmers reason informally in a local way. Data structure algorithms typically work by applying local surgeries that rearrange small parts of a data structure, such as rotating a small part of a tree or inserting a node into a list. Informal reasoning usually concentrates on the effects of these surgeries, without picturing the entire memory of a system. We summarize this local reasoning viewpoint as follows.

To understand how a program works, it should be possible for reasoning and specification to be confined to the cells that the program actually accesses. The value of any other cell will automatically remain unchanged.

Local reasoning is intimately tied to the complexity of specifications. Often, a program works with a circumscribed collection of resources, and it stands to reason that a specification should concentrate on just those resources that a program accesses. For example, a program that inserts an element into a linked list need know only about the cells in that list; there is no need (intuitively) to keep track of all other cells in memory when reasoning about the program.

The central idea of the approach studied in this paper is of a “spatial conjunction”  $P * Q$ , that asserts that  $P$  and  $Q$  hold for separate parts of a data structure. The conjunction provides a way to compose assertions that refer to different areas of memory, while retaining disjointness information for each of the conjuncts. The locality that this provides can be seen both on the level of atomic heap assignments and the level of compound operations or procedures. When an alteration to a single heap cell affects  $P$  in  $P * Q$ , then we know that it will not affect  $Q$ ; this gives us a way to short-circuit the need to check for potential aliases in  $Q$ . On a larger scale, a specification  $\{P\}C\{Q\}$  of a heap surgery can be extended using a rule that lets us infer  $\{P * R\}C\{Q * R\}$ , which expresses that additional heap cells remain unaltered. This enables the initial specification  $\{P\}C\{Q\}$  to concentrate on only the cells in the program’s footprint.

The basic idea of the spatial conjunction is implicit in early work of Burstall [3]. It was explicitly described by Reynolds in lectures in the fall of 1999; then an intuitionistic logic based on this idea was discovered independently by Reynolds [20] and by Ishtiaq and O’Hearn [7] (who also introduced a spatial implication  $P \multimap Q$ , based on the logic BI of bunched implications [11, 17]). In addition, Ishtiaq and O’Hearn devised a classical version of the logic that is more expressive than the intuitionistic version. In particular, it can express storage deallocation.

Subsequently, Reynolds extended the classical version by adding pointer arithmetic. This extension results in a model that is simpler and more general than our previous models, and opens up the possibility of verifying a wider range of low-level programs, including many whose properties are difficult to capture using type systems. Meanwhile, O’Hearn fleshed out the theme of local reasoning sketched in [7], and he and Yang developed a streamlined presentation of the logic based on what we call the “small axioms”.

In this joint paper we present the pointer arithmetic model and assertion language, with the streamlined Hoare logic. We illustrate the formalism using programs that work with a space-saving representation of doubly-linked lists, and a program that copies a tree.

Two points are worth stressing before continuing. First, by *local* we do not merely mean *compositional* reasoning: It is perfectly possible to be compositional and global (in the state) at the same time, as was the case in early denotational models of imperative languages. Second, some aspects of this work bear a strong similarity to semantic models of local state [19, 15, 16, 13, 12]. In particular, the conjunction  $*$  is related to interpretations of syntactic control of interference [18, 10, 12], and the Frame Rule described in Section 3 was inspired by the idea of the expansion of a command from [19, 15]. Nevertheless, local reasoning about state is not the same thing as reasoning about local state: We are proposing