Abstract:

Stacks, Heaps and Regions: One Logic to Bind Them

David Walker
Computer Science Department
Princeton University

Systems of proof-carrying code and typed assembly language guarantee a variety of important safety properties for low-level programs. However, implementers often spend the most time and effort creating mechanisms related to checking aliasing properties, data layout and memory management. One of the main causes of this phenomenon is simply the sheer number and complexity of different memory invariants aliasing relationships that are possible in a certifying compiler. In the face of this complexity, ad hoc techniques for ensuring memory safety will break down.

A promising new way to manage the complexity of memory management invariants in safe low-level systems is to use substructural logics. The expressive connectives of some substructural logics are able to capture the spatial orientation of a data structure in a concise fashion without having to rely upon the ad hoc auxiliary predicates needed by conventional logics. In this talk, we will explain how to develop a substructural logic with connectives capable of expressing a variety of spatial properties of data including:

- Juxtaposition of one piece of storage next to another
- Separation (disjointedness) of one piece of storage with respect to another
- Containment of one piece of storage within another

Great care has been taken to define each connective in our logic orthogonally to the others, so a particular proof-carrying code system can employ the exact fragment of logic that suits its needs. The connectives also compose nicely and allow us to express rich invariants involving data structures allocated on the stack, in the heap, or in user-managed memory regions. We have defined the syntactic proof rules for our logic, given a storage semantics to the connectives, and defined a typed assembly language that uses the logic to ensure safe stack, heap and region allocation and deallocation of data.

This is joint research with Amal Ahmed and Limin Jia.