Regions and Permissions for Data Invariants

Romain Bardou

14 January 2011

Introduction

we deal with complex data structures involving:

- pointers, with aliasing
- data invariants

maintaining invariants with pointer aliasing is hard as values could be modified unexpectedly

we propose a modular type system to control aliasing and invariants in a static way, i.e. with less proof obligations

Example of Complex Data Structure

hash tables:

- invariant: $(key, data)$ stored at index: $hash(key) \bmod len$
- keys must not be modified after having been inserted (invariant would be broken)
- associated values may be modified

memoize a function f using such hash tables:

- associate $f(x)$ to x
- invariant: if (x, y) is in the table, $y = f(x)$
- associated values must not change unexpectedly (invariant would be broken)

Related Works and Our Proposal

several dynamic approaches exist:

- Spec# ownership [Barnett et al 04]
- regional logic [Banerjee et al 08]
- characteristic formulas [Charguéraud 10]
- separation logic [Reynolds 02]
- dynamic frames [Kassios 06]

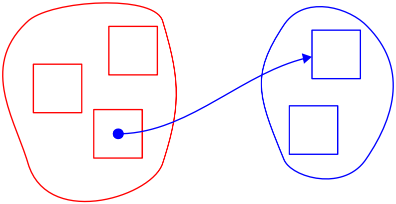
emphasis is on the logic

our proposal is static, i.e. "more automatic", and uses:

- regions [Tofte, Talpin, Jouvelot 91]
- with permissions [Crary et al 99, Charguéraud 07]

emphasis is on the type system

Pointers Belong to Regions



we keep track of the region of a pointer in its type

Permissions Give Information About Regions

permissions are linear information

- $\rho^{\emptyset}$: region ρ is currently empty
- ρ°: region ρ is singleton
- $\rho^{\times}$: region ρ is singleton and invariant holds
- ρ^{G}: region ρ is group and invariants hold
- ...

example:

fun add [ρ_t: *Hashtbl*, ρ: *Key*; ρ_d: *Data*](t: [ρ_t], k: [ρ], d: [ρ_d])
  consumes ${\rho_t}^{\times}$, $\rho^{\times}$
  produces ${\rho_t}^{\times}$

Ownership Tree Between Regions

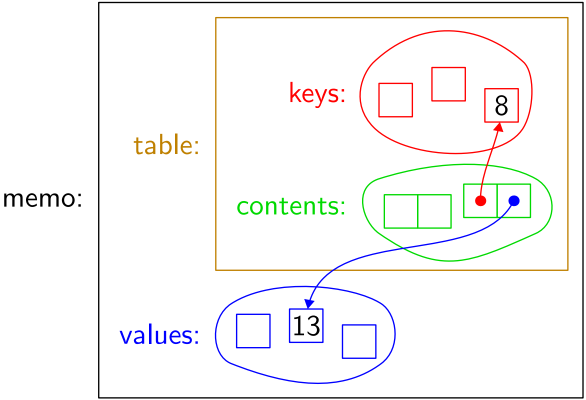
a region can own other regions

when ρ is packed ($\rho^{\times}$ or ρ^{G}), permissions on owned regions are unavailable:

- no one can modify owned regions when owner is packed
- allow modular reasoning: owned regions may be hidden

packing hides owned permissions, unpacking restores them

Ownership Tree Between Regions (Example)



pointers of region values are used inside table.contents but are not owned by the table but by the Fibonacci data structure

Soundness (1/2)

Definition 1 (heap coherence w.r.t. permissions)

- if $\rho^\times$ or ρ^G is available, invariants of pointers of ρ hold
- ...

Theorem 1 (coherence preservation) coherence is preserved through program reduction

Soundness (2/2)

Definition 2 (Why translation) we translate our programs to Why programs by encoding each region as a small, separate heap

Theorem 2 (Why translation) if translated program reduces (i.e. proof obligations are proven), original program reduces too

Conclusion

only proof obligations from our system: invariants when packing
moreover, regions are separated in proof obligations

implemented as a prototype tool called Capucine
[http://romain.bardou.fr/capucine]

we trade expressivity for staticity

- future works: extend the logic so the user can handle cases not handled by our type system