Aliasing control with view-based typestate

Jonathan Aldrich
Filipe Militão
Luís Caires

Carnegie Mellon University, Pittsburgh, Pennsylvania, USA
Faculdade de Ciências e Tecnologia - Universidade Nova de Lisboa, Lisboa, Portugal

Introduction

Traditional type systems model state implicitly.

For instance, in a File class, although it only makes sense for close to be available after a call to open, they are commonly merged together in the same class instead of distinguishing the states.

Typestate systems (such as Plural [1]) model state explicitly.

However, this creates the issue of tracking state changes across possibly aliased object references.

Pair Example (non overlapping and non interfering)

The system type can independently track the initialization state of the left and right fields through separate views.

```java
class Pair { 
    L : A; 
    R : B;
    // the empty (void) permission
    none init (<>none x, <>none y) (EmptyPair<>Pair); 
    this.setLeft(y); 
    this.setRight(x); 
} 
```

The permission to x will be taken. Thus afterwards only the empty permission on x remains.

```java
none setLeft (<>none x, EmptyLeft<>Left); this.l = x 
none setRight (<>none x, EmptyRight<>Right); this.r = x 
```

// initialized Pair

The permission to x will be borrowed and fully returned at the end of the method.

```java
int someOneR (<>R x) (Left<>Left); 
return x.getValue() + this.l.getValue(); 
this.r = unreachable inside someOneR (it is in a different view, not in left)
```

```java
none outsideRight (Right<>Right x) (Left<>Left) 
```

```java
none pair-method() (Pair<>Pair) 
```

Iterator Example (overlapping, but non interfering, unbounded sharing)

```java
class Iterator <<pretend there is always a next...>> 
```

Object next() {Iterator<>Iterator; ... ... }

```java
class Collection { 
    none add(Object<>none o) {Collection<>Collection; ... ... 
    int size() {UnderIteration<>UnderIteration; ... ... 
    Collection = UnderIteration * Iterable
    Collection = ? Iterable * ? Iterable
    size does not modify the collection
```

We employ a simplification of the idea of fractions (from Fractional Permissions [2]) to count the number of existing iterable_slices.

- represents a fixed fraction of 1/2 value. So in the beginning we have: Iterable = Iterable * Iterable
- it is used to represent some fixed number of items (or more), therefore for ? ? this second view equation allows for Iterable = Iterable * Iterable

(work in progress...)

Related Work

Future Work: overlapping and interfering sharing with coordination

Future Work: overlapping and interfering sharing with coordination

Research Questions

How to coordinate and type check sharing of overlapping data? (including other types of coordinations, such as multiple readers with a single writer, etc.)

Does knowing how each view uses its slice of a class help in checking race conditions, incorrect coordination (deadlocks), and lock/atomic blocks? (for instance, merging views that were used by different threads can expose non-isolation, etc.)