Monotonic State in 

Aseem Rastogi
Microsoft Research India

(Computer Aided Security Proofs, Aarhus, Denmark)

Example: Monotonic Counter

```
val alloc: unit    -> ST counter (…)
val read : counter -> ST int     (…)
val incr : counter -> ST unit    (…)
```

```
let x = read c in
assert (x > 2);

complex_stateful_procedure ();    //ST unit (fun _ -> True) (fun _ _ _ -> True)

let y = read c in
assert (y > 2)          //How can we verify such “stable” properties?
```

Stateful style is quite heavy

```
let x = read c in
assert (x > 2);

complex_stateful_procedure ();    //ST unit (fun _ -> True) (fun _ _ _ -> True)

let y = read c in
assert (y > 2)
```

Can carry such invariants in the specs of stateful functions (tedious)

```
(requires (fun h0 -> sel h0 c > 2 /\ …)) (ensures (fun _ _ h1 -> sel h1 c > 2 /\ …))
```

Instead …

```
val alloc: unit    -> ST counter (…)
val read : counter -> ST nat     (…)
val incr : counter -> ST unit    (…)
```

Counter only increases, so the property `(sel h c > 2)` is stable

Can we get such properties for free

Crypto Applications of Monotonic State

- Reasoning about idealized logs
- Modeling low-level, safe arrays
- …

Monotonic State in F*

Each reference is equipped with a preorder relation

```
type mref (a:Type) (rel:preorder a)    //rel is a reflexive and transitive relation

val write (#a:Type) (#rel:preorder a) (r:mref a rel) (x:a)
  :ST unit (requires (fun h0 -> r (sel h0 r) x /\ …)) (…)
```

Stable predicates w.r.t. preorders

```
type predicate (a:Type) = a -> Type

let stable (#a:Type) (p:predicate a) (rel:preorder a) =
  forall (x:a) (y:a). (p x /\ rel x y) => p y
```

Witnessing and Recalling Stable Predicates

A **New pure proposition** to bind a stable predicate to a reference

```
val token (#a:Type) (#rel:preorder a) (r:mref a rel) (p:predicate a{stable p}) :Type0
```

A pair of **New** stateful functions to witness and recall tokens

```
val witness_token #a #rel (r:mref a rel) (p:predicate a{stable p})
  :ST unit (fun h0      -> p (sel r h0))
           (fun h0 _ h1 -> h0 == h1 /\ token r p)

val recall_token #a #rel (r:mref a rel) (p:predicate a{stable p})
  :ST unit (fun h0      -> token r p)
           (fun h0 _ h1 -> h0 == h1 /\ p (sel r h0))
```

Metatheoretic argument for soundness

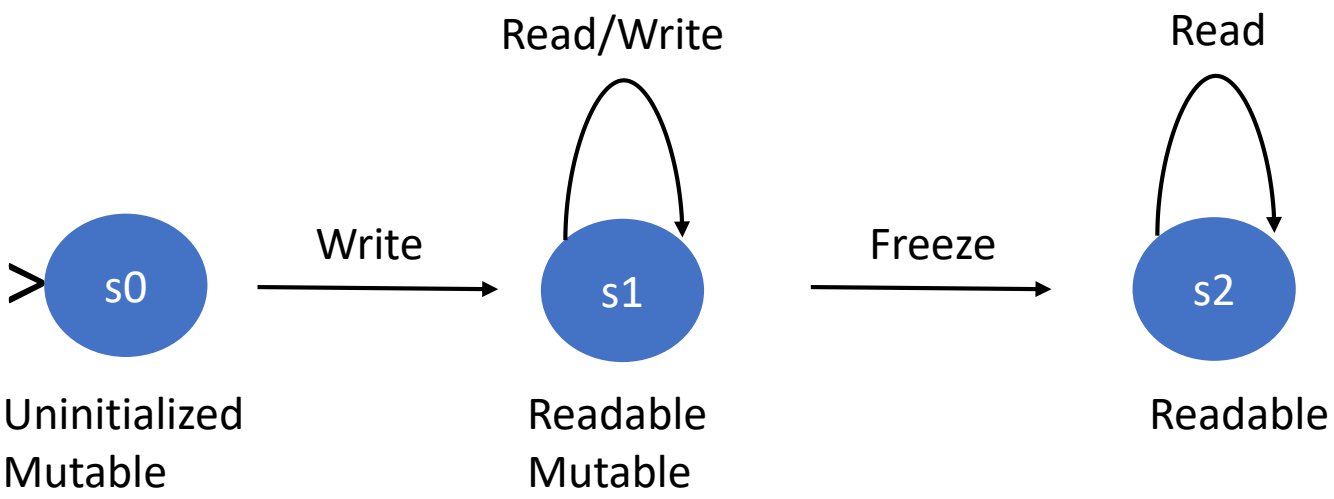
Ahman et al. POPL 2018 (to appear)

Monotonic Counter Example

(In emacs)

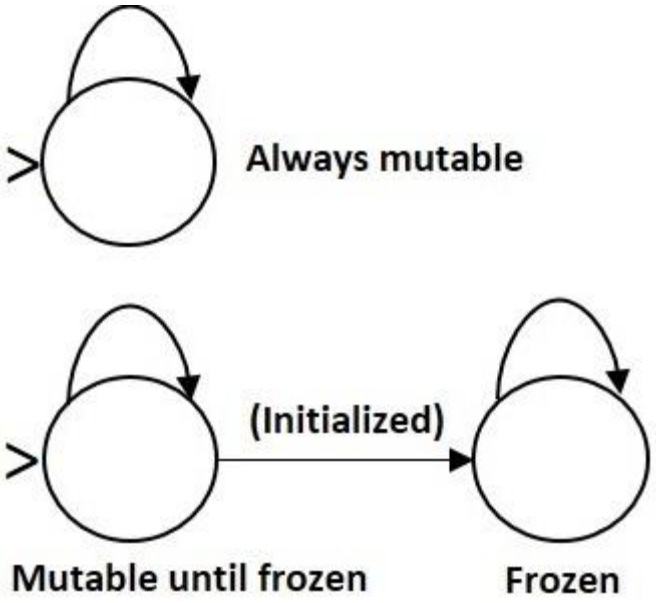
Refs with support for initialization and freeze

(In emacs)



Monotonic Arrays

(In emacs)



- Readability at initialized indices
- Freezing a fully initialized array
- Mutability is stateful