Discussion 17

Memoization

Kenneth Fang (kwf37), Newton Ni (cn279)

March 27, 2019

Key Ideas

- Hide mutable state under functional interface

Key Ideas

- Hide mutable state under functional interface
- Best of both worlds: fast and easy to reason about

Memoization

```
(** [memoize f] is a function [f’] that
  * mutably caches previously calculated results. *)
val memoize: (’a -> ’b) -> (’a -> ’b)
```

Memoization

- Can we apply `memoize` directly to recursive functions?

Memoization

- Can we apply `memoize` directly to recursive functions?
- `let rec factorial n =`
- `  if n = 0 then 1 else n * (factorial (n - 1))`

Memoization

- Can we apply `memoize` directly to recursive functions?
- `let rec factorial n =`
- `  if n = 0 then 1 else n * (factorial (n - 1))`
- `(memoize factorial) 5`

Recursive Memoization

- Need to adjust structure of recursive function

Recursive Memoization

- Need to adjust structure of recursive function
- Add self parameter to inject memoization

Recursive Memoization

- Need to adjust structure of recursive function
- Add self parameter to inject memoization
- Memoization all the way down

Recursive Memoization

```
let factorial self n =
  if n = 0 then 1 else n * (self (n - 1))
```

Recursive Memoization

```
let factorial self n =
  if n = 0 then 1 else n * (self (n - 1))

let memoize_rec f =
  failwith "Unimplemented"
```