First Order Functions

- Correspond to *computations*, not *values*.
- Cannot be stored in data structures or passed to/from functions.
- Do not use heap space for representation (no closures).
- Like functions in C.

Tracking Effects

- Allow side effects in functions.
- Use an “effect system” to track effects.
- HM+qualified types has the necessary machinery.

- New constants:

```
kind Prim = IO# *
          | * #-> Prim
```

- Example:

```
prim_put_char :: Char #-> IO# ()
              :: Prim
```

Effects with Qualified Types

- Using polymorphism and qualified types to collect effects:
- Instead of concrete effects like `IO#` use a predicate `CanIO`:

```
prim_put_char :: CanIO m => Char #-> m ()
```

- Need to keep track of the current "effect context".

Summary

- Exposes some of the inner workings of existing implementations.
- Language makes some promises about representation (i.e., no allocation).
- In low level code heap allocation could be an effect?
- Potentially useful for FFI purposes.