subst :: Expr, String, Value → Expr (Value ≡ Num)

interp :: FDs, Expr → Value

1. Seems slow

2. Keep source

3. Violates intuition of "lookup"

DEFER SUBSTITUTION

subst :: Expr, String, Value -> Expr

type Environment = StringDict<Value>
Env

DeferredSubst

```
fun lookup(env::Env, id::String) -> Value:
  cases(Option<Value>) env.get(id):
    | some(v) => v
    | none => raise(unbound-id)
  end
end
```

```
fun do-op(op::Op, v1::Value,
          v2::Value)
  -> Value
```

```
fun interp(fds::FDs, expr::Expr, env::Env) -> Value:
  cases(Expr) expr:
    | e-num(n) => n
    | e-op(op, l, r) => do-op(op, interp(fds, l, env),
                                  interp(fds, r, env))
    | e-if(c, thn, els) => ...
    | e-id(id) => lookup(env, id)
    | e-app(fname, arg-exp) =>
      cases(FD) lookup-fd(fds, fname):
        | fd(arg-name, body) =>
          av = interp(fds, arg-exp, env)
          interp(fds, body,
                 env.set(arg-name, av))
end
```

[dict: arg-name, av]

interp-env-fixed
interp-subst

$\stackrel{?}{=}$

interp-env

optimization

LEXICAL SCOPE

DYNAMIC SCOPE

```
fun f(x): x end
fun g(x): f(2) end
g(3)
```

[dict: "x", 2, ~~"x", 3~~]

[dict: "x", 3]

$$\frac{2}{S} \overset{!}{=} \frac{2}{E}$$

```
fun f(x): y end
fun g(y): f(2) end
g(3)
```

[dict: "x", 2, "y", 3]

[dict: "y", 3]

$$\frac{\text{unbound-id}}{S} \neq \frac{3}{E}$$