1. Guarantee about avoiding (certain) errors

2. Prediction of types of values computed

New obj errors:

field-not-found

not-an-object

Concerns:

equals on objects

fields w/ same name, different type

field's missing w/ same type

```
empty = letrec self = {
    is-member: lam(elt): false end
}
```

```
MT = {
    is-member :: (Any -> Bool)
}
```

```
link1 = letrec self={
    is-member: lam(elt):
        (self.first == elt) or self.rest.is-member(elt)
    end,
    first: 1,
    rest: empty
}
```

```
{
  is-member: (Any -> Bool),
  first: Num,
  rest: MT
}
```

```
type-of(o.x)
    OT = type-of(o)
    is OT an object type?
        does OT have x field?
            return OT.x
        else field-not-found
    else not-obj
```

```
(lam(x) x).fld
f = (lam(x) x)
F.fld
```

```
fun link(f :: Num, r :: TRest)

  letrec self = {
    is-member: lam(elt): (elt == self.first) or
                         self.rest.is-member(elt) end,

    first: f,
    rest: r ::TRest
  } ; self end
end
```

type-of(1) = Num → link(1, empty)

type-of(empty) = TRest link(1, link(2, empty))

type-of(link(2, empty)) = TRest

treat as

{is-member :: (Any → Bool)} ← {is-member :: (Any → Bool),
first :: Num,
rest :: TRest }

if type-of(e) = $\{f_1 :: \tau_1, \ldots, f_n :: \tau_n\}$

then type-of(e) = $\{f_1 :: \tau_1, \ldots, f_{n-1} :: \tau_{n-1}\}$

Drop a field

Another rule for reordering exists)

$\{x : 5, y : 10\}$ $\{x :: Num, y :: Num\}$

$\{x :: Num\}$

$\{\}$

SUBTYPING

"subtype of"

$\tau_1 <: \tau_2$

WIDTH SUBTYPING

$\{f_1 :: \tau_1, \ldots, f_n :: \tau_n\} <: \{f_1 :: \tau_1, \ldots, f_{n-1} :: \tau_{n-1}\}$

$\{f_1 :: \tau_1, f_2 :: \tau_2, \ldots, f_n :: \tau_n\} <: \{f_2 :: \tau_2, f_1 :: \tau_1, \ldots f_n :: \tau_n\}$

$\tau_1 <: \tau_2$ if $\tau_1 <: \tau_3$ $\tau_3 <: \tau_2$

$\{x :: Num\}$

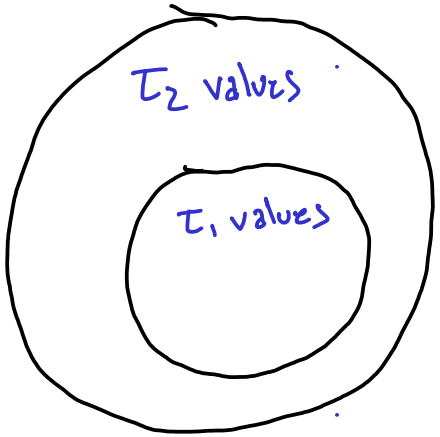
$\{x: 1\}$
$\{x: 2\}$
$\vdots$
$\{x: 3\}$

$\{x :: Num, y :: Num\}$

$\{x:1, y:1\}$
$\{x:2, y:1\}$
$\vdots$
$\{x:3, y:4\}$

have this type?

$T_1 <: T_2$



TYPESCRIPT

JAVASCRIPT

$<:$
$:\ T$