A Compiler for Beluga

Francisco Ferreira
McGill University

PhD Student

fferre8@cs.mcgill.ca

A versatile representation, powerful pattern matching compilation and dependent types

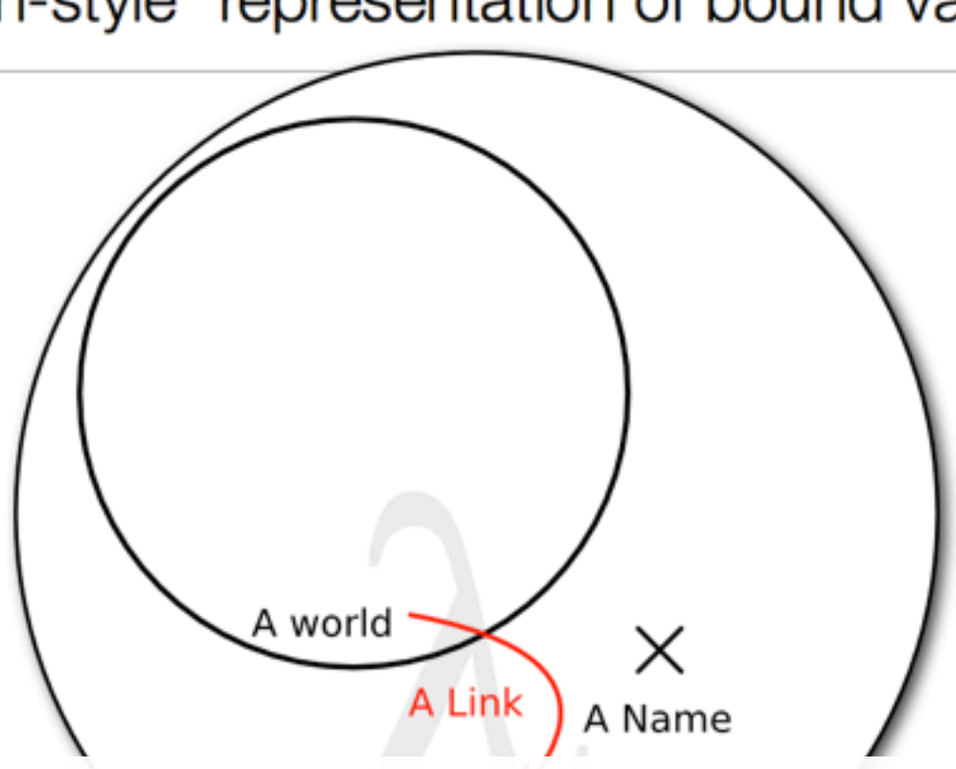
the normalization function

```
rec norm : {φ:ctx} (exp T) [φ] → (exp T) [φ] =
fn e ⇒ case e of
| ❶[φ] ❷#p… ⇒ [φ] #p…

| [φ] ❸lam (❹λx. M…x) ⇒
  let [φ, x:exp _] N…x = norm (
  [φ] lam λx. N…x

| [φ] app (M1…) (M2…) ⇒
  (case norm ([φ] M1…) of
   | [φ] lam (λx. M'…x) ⇒ norm
   | [φ] N1… ⇒
     let [φ] N2… = norm ([φ] M2
     [φ] app (N1…) (N2…)
  )
;
```

the “fresh-style” representation of bound variables



Compiling dependently typed programs with binders since... (we are just starting!)