unicaml

Programming in the untyped λ-calculus

Booleans

```
let ift = Abs("x",Var "x");;

let t = Abs("x",Abs("y",Var "x"));;

let f = Abs("x",Abs("y",Var "y"));;
```

Church numerals

```
let rec iter f x n = if n=0 then x else App(f,(iter f x (n-1)));;

let church n = Abs("f",Abs("x",iter (Var "f") (Var "x") n));;

print_string (string_of_term (church 0));;
print_string (string_of_term (church 1));;
print_string (string_of_term (church 2));;

let succ =
  Abs("n",Abs("f",Abs("x",
    App(Var "f",(App(App(Var "n",Var "f"),Var "x"))))));;

let iszero =
  Abs("n",Abs("x",Abs("y",
    App(App(Var "n",Abs("z",Var "y")),Var "x")))));;

reducefix (App(iszero,(church 0)));;
reducefix (App(iszero,(church 1)));;

let rec unchurch t = match t with
  Abs(f,Abs(x,t')) -> (match t' with
    Var x -> 0
  | App(Var f,s) -> 1 + unchurch (Abs(f,Abs(x,s)))
  | _ -> raise Error)
  | _ -> raise Error;;

unchurch (church 2);;

unchurch (reducefix (App(succ,(church 1))));;
```

Pairs

Pairs and their accessor functions, as well as tuples, can also be encoded in the untypedlambda|untyped lambda calculus. Yet, their type is much less informative than the type of primitive OCaml tuples.

```
# let untyped_true = fun x y -> x;;
val untyped_true : 'a -> 'b -> 'a = <fun>

# let untyped_false = fun x y -> y;;
val untyped_false : 'a -> 'b -> 'b = <fun>

# let untyped_pair = fun f s b -> b f s;;
val untyped_pair : 'a -> 'b -> ('a -> 'b -> 'c) -> 'c = <fun>

# let untyped_fst = fun p -> p untyped_true;;
val untyped_fst : (('a -> 'b -> 'a) -> 'c) -> 'c = <fun>

# let untyped_snd = fun p -> p untyped_false;;
val untyped_snd : (('a -> 'b -> 'b) -> 'c) -> 'c = <fun>

# untyped_fst (untyped_pair 1 "alice");;
- : int = 1

# untyped_snd (untyped_pair 1 "alice");;
- : string = "alice"
```

```
let pair = Abs("x",Abs("y",Abs("z",App(App(Var "z",Var "x"),Var "y"))));;
let fst = Abs("x",App(Var "x",t));;
let snd = Abs("x",App(Var "x",f));;

let t1 = App(snd,App(App(pair,church 0),church 1));;

unchurch (reducefix t1);;
```

Predecessor

```
let z = App(App(pair,church 0),church 0);;
let s = Abs("x",App(App(pair,App(snd,Var "x")),App(succ,App(snd,Var "x"))));;

unchurch (reducefix (App(snd,App(s,App(s,z)))));;

let pred = Abs("n",App(fst,App(App(Var "n",s),z)));;

unchurch (reducefix (App(pred,church 9)));;
```

Fixed points

```
let u = Abs("x",Abs("y",App(Var "y",App(App(Var "x",Var "x"),Var "y"))));;
let theta = App(u,u);;
```

Addition

```
let fadd = Abs("f",Abs("n",Abs("m",App(App(App(ift,App(iszero,Var "n")),Var "m"),App(succ,App(App(Var "f",App(pred,Var "n")),Var "m"))))));;

let add = App(theta,fadd);;

unchurch (reducefix (App(App(add,church 3),church 5)));;
```

---

untypedlambda-prog.txt · Last modified: 2015/10/08 15:20 (external edit)