Untyped λ-terms

```
module lcsig.

kind tm  type.
type ab  (tm -> tm) -> tm.
type app tm -> tm -> tm.

type example   string -> tm -> o.

example "I" (ab x\ x).
example "K" (ab x\ ab y\ x).
example "S" (ab x\ ab y\ ab z\ app (app x z) (app y z)).
example "self-apply" (ab x\ app x x).
example "0" (ab f\ ab x\ x).
example "1" (ab f\ ab x\ app f x).
example "2" (ab f\ ab x\ app f (app f x)).
example "mystery"
 (app (app (ab x\ ab y\ app (ab w\ app w x) y)
          (ab u\ app (ab w\ w) u))
      (app (ab x\ ab y\ app (ab w\ app w x) y)
          (app (ab u\ app (ab w\ w) u)
               (app (ab x\ ab y\ app (ab w\ app w x) y)
                    (ab u\ app (ab w\ w) u))))).
example "Y"
   (ab x\ app (ab y\ app x (app y y))
              (ab y\ app x (app y y))).
```

Evaluation

```
module eval.
accumulate lcsig.

% call-by-name evaluation

type cbn   tm -> tm -> o.

cbn (ab R) (ab R).
cbn (app M N) V :- cbn M (ab R), cbn (R N) V.

% call-by-value evaluation

type cbv   tm -> tm -> o.

cbv (ab R) (ab R).
cbv (app M N) V :- cbv M (ab R), cbv N U, cbv (R U) V.
```

Simple Type Inference

```
module typeinf.
accumulate lcsig.

kind ty   type.
type arr  ty -> ty -> ty.

type infer  tm -> ty -> o.

infer (app M N) B  :-
  infer M (arr A B), infer N A.

infer (ab R) (arr A B) :-
   pi x\ infer x A => infer (R x) B.
```

```
unix> terzo
Terzo lambda-Prolog, Version 1.1b, Built Fri Jun  6
[reading file /home/trantor.22/dale/.terzo.rc]  ....
Terzo> #load "lc.mod".
[reading file lc.mod] ...
Terzo> #query typeinf example N Tm, infer Tm Ty.
N = "I"
Tm = ab x\ x
Ty = arr B B ;

N = "K"
Tm = ab x1\ ab y\ x1
Ty = arr B1 (arr B2 B1) ;

N = "S"
Tm = ab x2\ ab y1\ ab z\ app (app x2 z) (app y1 z)
Ty = arr (arr B (arr A C)) (arr (arr B A) (arr B C)) ;

N = "0"
Tm = ab f\ ab x3\ x3
Ty = arr A1 (arr B5 B5) ;

N = "1"
Tm = ab f1\ ab x4\ app f1 x4
Ty = arr (arr A2 B6) (arr A2 B6) ;

N = "2"
Tm = ab f2\ ab x5\ app f2 (app f2 x5)
Ty = arr (arr A3 A3) (arr A3 A3) ;
```

```
N = "mystery"
Tm =
 app (app (ab x6\ ab y2\ app (ab w\ app w x6) y2)
          (ab u\ app (ab w1\ w1) u))
  (app (ab x7\ ab y3\ app (ab w2\ app w2 x7) y3)
    (app (ab u1\ app (ab w3\ w3) u1)
      (app (ab x8\ ab y4\ app (ab w4\ app w4 x8) y4)
           (ab u2\ app (ab w5\ w5) u2))))
Ty = arr (arr (arr B7 B7) Ty1) Ty1 ;

no more solutions
Terzo> #query eval.
?- example "mystery" Tm, cbn Tm V.
Tm =
 app (app (ab x\ ab y\ app (ab w\ app w x) y)
          (ab u\ app (ab w1\ w1) u))
  (app (ab x1\ ab y1\ app (ab w2\ app w2 x1) y1)
    (app (ab u1\ app (ab w3\ w3) u1)
      (app (ab x2\ ab y2\ app (ab w4\ app w4 x2) y2)
           (ab u2\ app (ab w5\ w5) u2))))
V = ab y\ app (ab w\ app w (ab u\ app (ab w\ w) u)) y

?- example "I" I, example "Y" Y, cbn (app Y I) V.
GC #0.0.0.1.3.41:   (32 ms)
GC #0.0.0.2.4.81:   (77 ms)
GC #0.0.1.3.5.115:   (99 ms)
GC #0.1.2.4.6.134:   (43 ms)
interrupt
Terzo>
```