bitbreeder

Claude Heiland-Allen

2013–2019

Contents

1 bitbreeder.hs 2
2 bitbreeder_video.hs 6
3 Compile.hs 7
4 Config.hs 8
5 Database.hs 8
6 debug.c 9
7 deps.sh 9
8 encode.sh 10
9 Evolve.hs 10
10 Expression.hs 12
11 expr.frag 12
12 extra/bitbreeder.cabal 13
13 extra/LICENSE 14
14 extra/Setup.hs 26
15 Genetics.hs 26
16 .gitignore 29
17 glyphs.png 30
18 go.c 30
19 gradient.ppm 31
20 judge.c 31
21 live.c 36
22 Makefile 38
23 Metric.hs 39
24 Population.hs 40
25 README 41
26 spectrogram.c 44
27 start.sh 48
28 Statistics.hs 49
29 stroke.frag 50
30 Video.hs 50

1 bitbreeder.hs

```
module Main(main) where

import Control.Exception (handle, SomeException)
import Control.Monad (forM_, when)
import Control.Monad.Random (evalRandIO)
import Control.Concurrent (forkIO) -- , threadDelay)
import GHC.Conc (numCapabilities)
import Data.Maybe (listToMaybe)
import Data.Time.Clock (getCurrentTime, diffUTCTime)
import Data.Vector.Unboxed ((!), (//))
import System.IO (hSetBuffering, BufferMode(LineBuffering), hPutStrLn, stdout) ↙
   ↳ -- , stderr)

import Graphics.UI.Gtk
import Control.Concurrent.STM
import System.Process

import Expression
import Evolve
import Compile
import Population
import qualified Metric as M

gui :: (TVar [TVar Population], TBQueue Item) -> IO ()
gui (dbsV, toAudio) = do
  wi <- windowNew
  nb <- notebookNew
  visibleV <- newTVarIO 0
  _ <- nb `on` switchPage $ atomically . writeTVar visibleV
  let addTab tabname = do
        db <- atomically $ (readTVar . head) =<< readTVar dbsV
        databaseV <- newTVarIO db
        targetV <- newTVarIO M.emptyTarget
        weightV <- newTVarIO M.emptyWeight
        atomically $ modifyTVar dbsV (databaseV:)
        v <- vBoxNew False 5
        let resort = do
              t <- readTVar targetV
              w <- readTVar weightV
              modifyTVar databaseV $ target t w
            tSlider k = do
              t <- hScaleNewWithRange (-5) 5 0.01
              scaleSetDrawValue t False
              widgetSetSizeRequest t 512 24
              rangeSetValue t 0
              _ <- t `on` valueChanged $ do
                u <- rangeGetValue t
                atomically $ do
                  modifyTVar targetV (\(M.T z) -> M.T $ z // [(k, realToFrac u)↙
                     ↳ ])
                  resort
              return t
            wSlider k = do
              w <- hScaleNewWithRange 0 1 0.001
              scaleSetDrawValue w False
              widgetSetSizeRequest w 256 24
              rangeSetValue w 0
              _ <- w `on` valueChanged $ do
                u <- rangeGetValue w
                atomically $ do
                  modifyTVar weightV (\(M.W z) -> M.W $ z // [(k, realToFrac u)↙
                     ↳ ])
                  resort
              return w
            row k n = do
              h <- hBoxNew False 5
              l <- labelNew (Just n)
              widgetSetSizeRequest l 160 24
              t <- tSlider k
              w <- wSlider k
              boxPackStart h l PackNatural 0
              boxPackEnd h w PackGrow 0
              boxPackEnd h t PackGrow 0
              boxPackStart v h PackNatural 0
        forM_ (zip [0..] names) $ uncurry row
        widgetShowAll v
        page <- notebookAppendPage nb v tabname
        notebookSetCurrentPage nb page
        let topWatcher n = do
              n' <- atomically $ do
                mi <- listToMaybe . toAscList <$> readTVar databaseV
                case mi of
                  Just i | itemID i /= n -> do
                    vis <- (page ==) <$> readTVar visibleV
                    when vis $ writeTBQueue toAudio i
                    return (itemID i)
                  _ -> retry
              topWatcher n'
            visWatcher False = do
              atomically $ do
                vis <- (page ==) <$> readTVar visibleV
                if not vis then retry else do
                  mi <- listToMaybe . toAscList <$> readTVar databaseV
                  case mi of
                    Just i -> writeTBQueue toAudio i
                    _ -> return ()
              visWatcher True
            visWatcher True = do
              atomically $ do
                vis <- (page ==) <$> readTVar visibleV
                when vis retry
              visWatcher False
        _ <- forkIO $ topWatcher (-1)
        _ <- forkIO $ visWatcher False
        return ()
  namesV <- newTVarIO $ words "aardvark beaver chimp donkey elephant frog goat halibut iguana jackdaw kitten leopard manatee newt otter pigeon quail rabbit stoat tiger uncle velociraptor whale xtinct yow zzz"
  let addTab' = do
        name <- atomically $ do
          ~(n:ns) <- readTVar namesV
          writeTVar namesV ns
          return n
        addTab name
  b <- buttonNewFromStock stockAdd
  _ <- b `on` buttonActivated $ addTab'
  widgetShowAll b
  notebookSetActionWidget nb b PackStart
  set nb [notebookScrollable := True, notebookHomogeneous := True]
  set wi [windowTitle := "BitBreeder", containerChild := nb]
  widgetSetSizeRequest wi 1024 576
  windowMove wi 0 0
  _ <- wi `onDestroy` mainQuit
  widgetShowAll wi
  addTab'
  mainGUI

names :: [String]
names = [ m ++ " (" ++ p ++ ")" | m <- measurements, p <- parameters ] ++ ["novelty"]

measurements :: [String]
measurements =
  [ "loudness"
  , "tonality"
  , "centroid"
  , "variance"
  , "skewness"
  , "kurtosis"
  ]

parameters :: [String]
parameters =
  [ "average"
  , "variability"
  , "granularity"
  ]

main :: IO ()
main = do
  hSetBuffering stdout LineBuffering
  args <- initGUI
  gui =<< evolution args

breeder :: TVar [TVar Population] -> TBQueue E -> IO ()
breeder dbsV toJudge = loop
  where
    loop = do
      ess <- map (map itemExpr . toAscList) <$> atomically (mapM readTVar =<< readTVar dbsV)
      es <- evalRandIO (crossBreed ess)
      forM_ es $ atomically . writeTBQueue toJudge
      loop

judge :: TVar [TVar Population] -> TBQueue E -> Int -> Int -> IO ()
judge dbsV toJudge inc = loop
  where
    loop i = do
      e <- atomically $ readTBQueue toJudge
      ignoreErrors $ do
        let so = "./o/" ++ show i ++ ".so"
        compileSO e so
        (_, Just hout, _, p) <- createProcess (proc "./bitbreeder_judge" [so, show i]){ std_out = CreatePipe, std_err = Inherit }
        v@(M.A vv) <- M.read hout
        _ <- waitForProcess p
        when (vv ! 0 > 0) $ do
          let it = Item i e v
          atomically $ do
            dbs <- readTVar dbsV
            forM_ dbs $ \db -> modifyTVar db (update it)
      loop (i + inc)

ignoreErrors :: IO () -> IO ()
ignoreErrors = handle ((\_ -> return ()) :: SomeException -> IO ())

evolution :: [String] -> IO (TVar [TVar Population], TBQueue Item)
evolution args = do
  toJudge <- newTBQueueIO (fromIntegral $ 2 * numCapabilities)
  toAudio <- newTBQueueIO (fromIntegral $ 2 * numCapabilities)
  dbV <- newTVarIO empty
  dbsV <- newTVarIO [dbV]
  _ <- forkIO $ breeder dbsV toJudge
  _ <- forkIO $ audio args toAudio
  forM_ [0 .. numCapabilities - 1] $ forkIO . judge dbsV toJudge numCapabilities
  return (dbsV, toAudio)

audio :: [String] -> TBQueue Item -> IO ()
audio args toAudio = do
  (Just audioh, _, _, _) <- createProcess (proc "./bitbreeder_audio" []){ std_in↙
      ↳ = CreatePipe, std_err = Inherit }
  (Just videoh, _, _, _) <- createProcess (proc "./bitbreeder_video" args){ ↙
      ↳ std_in = CreatePipe, std_err = Inherit }
  hSetBuffering audioh LineBuffering
  hSetBuffering videoh LineBuffering
  hPutStrLn videoh (show (0 :: Double, I 0))
  start <- getCurrentTime
  let loop n = do
        it <- atomically $ readTBQueue toAudio
        when (itemID it /= n) $ do
          now <- getCurrentTime
          let frame :: Double
              frame = 25 * realToFrac (diffUTCTime now start + 4.5)
          hPutStrLn audioh ("./o/" ++ show (itemID it) ++ ".so")
          hPutStrLn videoh (show (frame, itemExpr it))
        loop (itemID it)
  loop (-1)
```

2 bitbreeder_video.hs

```
module Main(main) where

import Prelude hiding (init)

import Control.Exception (handle, SomeException)
import Control.Monad (forever, when)
import System.Environment (getArgs)
import System.IO (hSetBuffering, BufferMode(LineBuffering), stdin, stdout)

import Graphics.UI.GLFW

import Config (videoX, videoY, videoW, videoH)
import Expression
import Video

```

```
main :: IO ()
main = do
  hSetBuffering stdin LineBuffering
  hSetBuffering stdout LineBuffering
  args <- getArgs
  let record = ”--record” ‘elem‘ args
  True <- init
  windowHint $ WindowHint'Resizable False
  windowHint $ WindowHint'Decorated False
  Just window <- createWindow videoW videoH ”BitBreeder Expression” Nothing ↙
      ↳ Nothing
  setWindowPos window videoX videoY
  makeContextCurrent (Just window)
  s <- setupGL
  draw s (I 0)
  handle ((\_ -> return ()) :: SomeException -> IO ()) $ forever $ do
    ne@(n, e) <- readLn :: IO (Double, E)
    print ne
    draw s e
    swapBuffers window
    when record $ captureToPNG (pngFilename (floor n))
    pollEvents
  destroyWindow window
  terminate
```

3 Compile.hs

```
module Compile (compile, compileSO) where

import System.Process (rawSystem)

import Expression

compile :: E -> String
compile X = ”t”
compile (I i) = show i
compile (U u e) = compileU u (”(” ++ compile e ++ ”)”)
compile (B b e f) = compileB b (”(” ++ compile e ++ ”)”) (”(” ++ compile f ++ ”)↙
    ↳ ”)
compile (T e f g) = ”(” ++ compile e ++ ”)?(” ++ compile f ++ ”):(” ++ compile g↙
    ↳ ++ ”)”

compileU :: U -> String -> String
compileU Neg s = ”-” ++ s
compileU LNot s = ”-(!” ++ s ++ ”)”
compileU BNot s = ”˜” ++ s

compileB :: B -> String -> String -> String
compileB Add = op ”+”
compileB Sub = op ”-”
compileB Mul = op ”*”
compileB Div = fn ”safe_div”
compileB Mod = fn ”safe_mod”
compileB BAnd = op ”&”
compileB LAnd = \l r -> ”-(” ++ op ”&&” l r ++ ”)”
compileB BOr = op ”|”
compileB LOr = \l r -> ”-(” ++ op ”||” l r ++ ”)”
compileB XOr = op ”ˆ”
compileB ShL = op ”<<”
compileB ShR = op ”>>”
compileB Lt  = \l r -> ”-(” ++ op ”<” l r ++ ”)”
compileB Gt  = \l r -> ”-(” ++ op ”>” l r ++ ”)”

op, fn :: String -> String -> String -> String
op o a b = a ++ o ++ b
fn f a b = f ++ ”(” ++ a ++ ”,” ++ b ++ ”)”

compileSO :: E -> FilePath -> IO ()
compileSO e so = do
  let code = compile e
  _ <- rawSystem ”gcc”
          [ ”-std=c99”, ”-w”, ”-O3”, ”-shared”, ”-fPIC”, ”go.c”
          , ”-o”, so, ”-DT=” ++ code, ”-DCODE=\”” ++ code ++ ”\””
          ]
  return ()
```

4 Config.hs

```
module Config where

videoW, videoH, videoX, videoY :: Int

videoW = 1920
videoH = 540

videoX = screen1W - videoW
videoY = 0

{-
videoW = screen2W
videoH = (9 * videoW) `div` 16

videoX = screen1W
videoY = (screen2H - videoH) `div` 2
-}

screen1W, screen1H, screen2W, screen2H :: Int

screen1W = 1920
screen1H = 1080

screen2W = 1024
screen2H = 768
```

5 Database.hs

```
module Database(DB(), empty, insert, sortOn, toAscList, splitAt, fromList) where

import Prelude hiding (splitAt)

import Data.List (insertBy, sortBy)
import qualified Data.List as L (splitAt)
import Data.Ord (comparing)
```

```
data DB a = DB
  { _insert     :: a -> DB a
  , _sortOn     :: (a -> Double) -> DB a
  , _toAscList :: [a]
  , _splitAt :: Int -> (DB a, DB a)
  }

empty :: DB a
empty = mkDB (const 0) []

insert :: a -> DB a -> DB a
insert = flip _insert

sortOn :: (a -> Double) -> DB a -> DB a
sortOn = flip _sortOn

toAscList :: DB a -> [a]
toAscList = _toAscList

splitAt :: Int -> DB a -> (DB a, DB a)
splitAt = flip _splitAt

fromList :: [a] -> DB a
fromList = foldr insert empty

-- invariants
--   list == L.sortBy (comparing snd) list
--   all [metric x == y | (x, y) <- list]
mkDB :: (a -> Double) -> [(a, Double)] -> DB a
mkDB metric list = DB
  { _insert = \item     -> mkDB metric  (insertBy (comparing snd) (item, metric item) list)
  , _sortOn = \metric' -> mkDB metric' (sortBy   (comparing snd) (map (\(x, _) -> (x, metric' x)) list))
  , _toAscList = map fst list
  , _splitAt = \n -> let (lo, hi) = L.splitAt n list in (mkDB metric lo, mkDB metric hi)
  }
```

6 debug.c

```
#include <stdio.h>
#include <dlfcn.h>

int main(int argc, char **argv) {
  void *dl = dlopen(argv[1], RTLD_NOW);
  const char *code = dlsym(dl, "code");
  printf("%s\n", code);
  return 0;
}
```

7 deps.sh

```
#!/bin/sh
cabal sandbox init
cabal install alex happy
cabal install gtk2hs-buildtools
cabal install GLFW-b gtk MonadRandom OpenGLRaw stm syb syz Vector
```

8 encode.sh

```
#!/bin/bash
SESSION="${1}"
if [ "x${SESSION}" = "x" ]
then
  exit
fi
time ./bitbreeder_video --record < "${SESSION}.out" > /dev/null
FRAMES="$(( $(avprobe -v quiet -show_streams -i "${SESSION}.wav" -of json | grep↙
    ↳ duration_ts | sed 's|.*: \(.*\),.*|\1|g') / 1920 ))"
pushd "${SESSION}"
PREVFRAME="00000000.png"
for FRAME in $(seq "$(( FRAMES - 125 ))")
do
  THISFRAME="$(printf "%08d" "${FRAME}").png"
  if [ -f "${THISFRAME}" ]
  then
    PREVFRAME="${THISFRAME}"
  else
    ln -s "${PREVFRAME}" "${THISFRAME}"
  fi
done
for FRAME in $(seq "$(( FRAMES - 124 ))" "${FRAMES}")
do
  THISFRAME="$(printf "%08d" "${FRAME}").png"
  ln -s "00000000.png" "${THISFRAME}"
done
popd
avconv -i "${SESSION}/%08d.png" -i "${SESSION}.wav" -shortest "${SESSION}.mkv"
```

9 Evolve.hs

```haskell
module Evolve (crossBreed) where

import Control.Monad (replicateM, forM)
import Control.Monad.Random (MonadRandom, getRandomR)

import Expression
import Genetics (nodes, exchange)

mutateI :: (Applicative m, MonadRandom m) => E -> m E
mutateI X = return X
mutateI (I i) = do
  k <- coin 0.1
  if k
    then do
      j <- getRandomR (1, 64)
      return (I j)
    else return (I i)
mutateI (U u e) = U u <$> mutateI e
mutateI (B b e f) = B b <$> mutateI e <*> mutateI f
mutateI (T e f g) = T <$> mutateI e <*> mutateI f <*> mutateI g

coin :: (Functor m, MonadRandom m) => Double -> m Bool
coin p = (< p) <$> getRandomR (0, 1)

terminal :: (Functor m, MonadRandom m) => m E
terminal = do
  c <- coin 0.5
  if c then return X else I <$> getRandomR (1, 64)

data F
  = FU U
  | FB B
  | FT
  deriving (Read, Show, Eq)

getRandomE :: (Functor m, MonadRandom m, Enum e, Bounded e) => m e
getRandomE = self
  where
    self = do
      mi <- return minBound `asTypeOf` self
      ma <- return maxBound `asTypeOf` self
      toEnum <$> getRandomR (fromEnum mi, fromEnum ma)

function :: (Functor m, MonadRandom m) => m F
function = do
  c <- coin 0.05
  if c then FU <$> getRandomE else do
    d <- coin 0.05
    if d then return FT else FB <$> getRandomE

grow :: (Applicative m, MonadRandom m) => Int -> m E
grow 0 = terminal
grow d = do
  c <- coin 0.25
  if c then terminal else do
    f <- function
    case f of
      FU u -> U u <$> grow (d - 1)
      FB b -> B b <$> grow (d - 1) <*> grow (d - 1)
      FT   -> T   <$> grow (d - 1) <*> grow (d - 1) <*> grow (d - 1)

breed :: (Applicative m, MonadRandom m) => E -> E -> m [E]
breed e0 e1 = do
  n0 <- getRandomR (0, nodes e0 - 1)
  n1 <- getRandomR (0, nodes e1 - 1)
  let (f0, f1) = exchange e0 n0 e1 n1
  return [f0, f1]

crossBreed :: (Applicative m, MonadRandom m) => [[E]] -> m [E]
crossBreed ess = do
  ws <- replicateM 2 $ getRandomR (0, length ess - 1)
  ~[e0, e1] <- forM ws $ \w -> do
    let es = take minPopCount (ess !! w)
    if length es < minPopCount then grow 5 else do
      n <- getRandomR (0, minPopCount - 1)
      mutateI (es !! n)
  e2 <- grow 5
  (e2:) <$> breed e0 e1
```

```
minPopCount :: Int
minPopCount = 64
```

10 Expression.hs

```
{-# LANGUAGE DeriveDataTypeable #-}
module Expression where

import Data.Data (Data)
import Data.Typeable (Typeable)

data E = X | I Int | U U E | B B E E | T E E E
  deriving (Read, Show, Eq, Ord, Data, Typeable)

data U = Neg | LNot | BNot
  deriving (Read, Show, Eq, Ord, Enum, Bounded, Data, Typeable)

data B = Add | Sub | Mul | Div | Mod | BAnd | LAnd | BOr | LOr | XOr | ShL | ShR | Lt | Gt
  deriving (Read, Show, Eq, Ord, Enum, Bounded, Data, Typeable)

count :: Integer -> Integer
count 0 = 0
count 1 = 65
count n = 3 * count (n - 1) + sum [ 14 * (count l + count r) | l <- [1 .. n - 2], let r = n - 1 - l ]
```

11 expr.frag

```
#version 400 compatibility

uniform sampler2D     glyphs;
uniform sampler2DRect expression;

const float yf  = 4.0/6.0;
const float ylo = 1.0/6.0;
const float yhi = 5.0/6.0;

float myTextureQueryLod(sampler2D tex, vec2 tc) {
  return textureQueryLod(tex, tc).y;
  //return max(0.0, 10.0 + log2(max(length(dFdx(tc)), length(dFdy(tc)))));
}

void main() {
  vec2 tc = gl_TexCoord[0].xy * vec2(0.125, 0.25 * yf);
  float lod = myTextureQueryLod(glyphs, tc);
  vec2 gc = floor(gl_TexCoord[0].xy * vec2(1.0, yf));
  vec3 glyph = texture2DRect(expression, gc).xyz;
  vec2 glyphCoord = glyph.xy;
  vec2 subCoord = fract(gl_TexCoord[0].xy * vec2(1.0, yf));
  if (subCoord.y < ylo) {
    glyphCoord = vec2(0.125, 0.75);
  } else if (yhi < subCoord.y) {
    glyphCoord = vec2(0.125, 0.75);
    subCoord.y += ylo - yhi;
  }
  subCoord.y -= ylo;
  subCoord *= vec2(0.125, 0.25 / yf);
  vec3 glyphRGB = textureLod(glyphs, glyphCoord + subCoord, lod).xyz;
  gl_FragColor = vec4(glyphRGB, 1.0);
}
```

12 extra/bitbreeder.cabal

```
name:                bitbreeder
version:             0.1.0.0
synopsis:            evolve noisy arithmetic expressions
-- description:
homepage:            http://code.mathr.co.uk/bitbreeder
license:             GPL-3
license-file:        LICENSE
author:              Claude Heiland-Allen
maintainer:          claude@mathr.co.uk
category:            Sound
build-type:          Simple
cabal-version:       >=1.8

executable bitbreeder
  main-is:
    bitbreeder.hs
  other-modules:
    Compile
    Database
    Evolve
    Expression
    Genetics
    Metric
    Population
  build-depends:
    base < 5,
    MonadRandom,
    gtk,
    process,
    stm,
    syb,
    syz,
    time,
    vector

executable bitbreeder_video
  main-is:
    bitbreeder_video.hs
  other-modules:
    Expression
    Video
  build-depends:
    base < 5,
    cairo,
    gtk,
    gtkglext,
    OpenGLRaw

--executable bitbreeder_judge
--  c-sources:
```

```
--      judge.c
--    extra-libraries:
--      m, dl, fftw3f

--executable bitbreeder_audio
--    c-sources:
--      live.c
--    extra-libraries:
--      m, dl, jack
```

13 extra/LICENSE

```
                    GNU GENERAL PUBLIC LICENSE
                       Version 3, 29 June 2007

 Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

                            Preamble

  The GNU General Public License is a free, copyleft license for
software and other kinds of works.

  The licenses for most software and other practical works are designed
to take away your freedom to share and change the works.  By contrast,
the GNU General Public License is intended to guarantee your freedom to
share and change all versions of a program--to make sure it remains free
software for all its users.  We, the Free Software Foundation, use the
GNU General Public License for most of our software; it applies also to
any other work released this way by its authors.  You can apply it to
your programs, too.

  When we speak of free software, we are referring to freedom, not
price.  Our General Public Licenses are designed to make sure that you
have the freedom to distribute copies of free software (and charge for
them if you wish), that you receive source code or can get it if you
want it, that you can change the software or use pieces of it in new
free programs, and that you know you can do these things.

  To protect your rights, we need to prevent others from denying you
these rights or asking you to surrender the rights.  Therefore, you have
certain responsibilities if you distribute copies of the software, or if
you modify it: responsibilities to respect the freedom of others.

  For example, if you distribute copies of such a program, whether
gratis or for a fee, you must pass on to the recipients the same
freedoms that you received.  You must make sure that they, too, receive
or can get the source code.  And you must show them these terms so they
know their rights.

  Developers that use the GNU GPL protect your rights with two steps:
(1) assert copyright on the software, and (2) offer you this License
giving you legal permission to copy, distribute and/or modify it.

  For the developers' and authors' protection, the GPL clearly explains
that there is no warranty for this free software.  For both users' and
```

```
authors' sake, the GPL requires that modified versions be marked as
changed, so that their problems will not be attributed erroneously to
authors of previous versions.

  Some devices are designed to deny users access to install or run
modified versions of the software inside them, although the manufacturer
can do so.  This is fundamentally incompatible with the aim of
protecting users' freedom to change the software.  The systematic
pattern of such abuse occurs in the area of products for individuals to
use, which is precisely where it is most unacceptable.  Therefore, we
have designed this version of the GPL to prohibit the practice for those
products.  If such problems arise substantially in other domains, we
stand ready to extend this provision to those domains in future versions
of the GPL, as needed to protect the freedom of users.

  Finally, every program is threatened constantly by software patents.
States should not allow patents to restrict development and use of
software on general-purpose computers, but in those that do, we wish to
avoid the special danger that patents applied to a free program could
make it effectively proprietary.  To prevent this, the GPL assures that
patents cannot be used to render the program non-free.

  The precise terms and conditions for copying, distribution and
modification follow.

                       TERMS AND CONDITIONS

  0. Definitions.

  ”This License” refers to version 3 of the GNU General Public License.

  ”Copyright” also means copyright-like laws that apply to other kinds of
works, such as semiconductor masks.

  ”The Program” refers to any copyrightable work licensed under this
License.  Each licensee is addressed as ”you”.  ”Licensees” and
”recipients” may be individuals or organizations.

  To ”modify” a work means to copy from or adapt all or part of the work
in a fashion requiring copyright permission, other than the making of an
exact copy.  The resulting work is called a ”modified version” of the
earlier work or a work ”based on” the earlier work.

  A ”covered work” means either the unmodified Program or a work based
on the Program.

  To ”propagate” a work means to do anything with it that, without
permission, would make you directly or secondarily liable for
infringement under applicable copyright law, except executing it on a
computer or modifying a private copy.  Propagation includes copying,
distribution (with or without modification), making available to the
public, and in some countries other activities as well.

  To ”convey” a work means any kind of propagation that enables other
parties to make or receive copies.  Mere interaction with a user through
a computer network, with no transfer of a copy, is not conveying.
```

```
  An interactive user interface displays ”Appropriate Legal Notices”
to the extent that it includes a convenient and prominently visible
feature that (1) displays an appropriate copyright notice, and (2)
tells the user that there is no warranty for the work (except to the
extent that warranties are provided), that licensees may convey the
work under this License, and how to view a copy of this License.  If
the interface presents a list of user commands or options, such as a
menu, a prominent item in the list meets this criterion.

  1. Source Code.

  The ”source code” for a work means the preferred form of the work
for making modifications to it.  ”Object code” means any non-source
form of a work.

  A ”Standard Interface” means an interface that either is an official
standard defined by a recognized standards body, or, in the case of
interfaces specified for a particular programming language, one that
is widely used among developers working in that language.

  The ”System Libraries” of an executable work include anything, other
than the work as a whole, that (a) is included in the normal form of
packaging a Major Component, but which is not part of that Major
Component, and (b) serves only to enable use of the work with that
Major Component, or to implement a Standard Interface for which an
implementation is available to the public in source code form.  A
”Major Component”, in this context, means a major essential component
(kernel, window system, and so on) of the specific operating system
(if any) on which the executable work runs, or a compiler used to
produce the work, or an object code interpreter used to run it.

  The ”Corresponding Source” for a work in object code form means all
the source code needed to generate, install, and (for an executable
work) run the object code and to modify the work, including scripts to
control those activities.  However, it does not include the work's
System Libraries, or general-purpose tools or generally available free
programs which are used unmodified in performing those activities but
which are not part of the work.  For example, Corresponding Source
includes interface definition files associated with source files for
the work, and the source code for shared libraries and dynamically
linked subprograms that the work is specifically designed to require,
such as by intimate data communication or control flow between those
subprograms and other parts of the work.

  The Corresponding Source need not include anything that users
can regenerate automatically from other parts of the Corresponding
Source.

  The Corresponding Source for a work in source code form is that
same work.

  2. Basic Permissions.

  All rights granted under this License are granted for the term of
copyright on the Program, and are irrevocable provided the stated
conditions are met.  This License explicitly affirms your unlimited
permission to run the unmodified Program.  The output from running a
```

covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

```
    b) The work must carry prominent notices stating that it is
    released under this License and any conditions added under section
    7.  This requirement modifies the requirement in section 4 to
    "keep intact all notices".

    c) You must license the entire work, as a whole, under this
    License to anyone who comes into possession of a copy.  This
    License will therefore apply, along with any applicable section 7
    additional terms, to the whole of the work, and all its parts,
    regardless of how they are packaged.  This License gives no
    permission to license the work in any other way, but it does not
    invalidate such permission if you have separately received it.

    d) If the work has interactive user interfaces, each must display
    Appropriate Legal Notices; however, if the Program has interactive
    interfaces that do not display Appropriate Legal Notices, your
    work need not make them do so.

  A compilation of a covered work with other separate and independent
works, which are not by their nature extensions of the covered work,
and which are not combined with it such as to form a larger program,
in or on a volume of a storage or distribution medium, is called an
"aggregate" if the compilation and its resulting copyright are not
used to limit the access or legal rights of the compilation's users
beyond what the individual works permit.  Inclusion of a covered work
in an aggregate does not cause this License to apply to the other
parts of the aggregate.

  6. Conveying Non-Source Forms.

  You may convey a covered work in object code form under the terms
of sections 4 and 5, provided that you also convey the
machine-readable Corresponding Source under the terms of this License,
in one of these ways:

    a) Convey the object code in, or embodied in, a physical product
    (including a physical distribution medium), accompanied by the
    Corresponding Source fixed on a durable physical medium
    customarily used for software interchange.

    b) Convey the object code in, or embodied in, a physical product
    (including a physical distribution medium), accompanied by a
    written offer, valid for at least three years and valid for as
    long as you offer spare parts or customer support for that product
    model, to give anyone who possesses the object code either (1) a
    copy of the Corresponding Source for all the software in the
    product that is covered by this License, on a durable physical
    medium customarily used for software interchange, for a price no
    more than your reasonable cost of physically performing this
    conveying of source, or (2) access to copy the
    Corresponding Source from a network server at no charge.

    c) Convey individual copies of the object code with a copy of the
    written offer to provide the Corresponding Source.  This
    alternative is allowed only occasionally and noncommercially, and
    only if you received the object code with such an offer, in accord
    with subsection 6b.
```

d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates

```
for a work that has been modified or installed by the recipient, or for
the User Product in which it has been modified or installed.  Access to a
network may be denied when the modification itself materially and
adversely affects the operation of the network or violates the rules and
protocols for communication across the network.

  Corresponding Source conveyed, and Installation Information provided,
in accord with this section must be in a format that is publicly
documented (and with an implementation available to the public in
source code form), and must require no special password or key for
unpacking, reading or copying.

  7. Additional Terms.

  ”Additional permissions” are terms that supplement the terms of this
License by making exceptions from one or more of its conditions.
Additional permissions that are applicable to the entire Program shall
be treated as though they were included in this License, to the extent
that they are valid under applicable law.  If additional permissions
apply only to part of the Program, that part may be used separately
under those permissions, but the entire Program remains governed by
this License without regard to the additional permissions.

  When you convey a copy of a covered work, you may at your option
remove any additional permissions from that copy, or from any part of
it.  (Additional permissions may be written to require their own
removal in certain cases when you modify the work.)  You may place
additional permissions on material, added by you to a covered work,
for which you have or can give appropriate copyright permission.

  Notwithstanding any other provision of this License, for material you
add to a covered work, you may (if authorized by the copyright holders of
that material) supplement the terms of this License with terms:

    a) Disclaiming warranty or limiting liability differently from the
    terms of sections 15 and 16 of this License; or

    b) Requiring preservation of specified reasonable legal notices or
    author attributions in that material or in the Appropriate Legal
    Notices displayed by works containing it; or

    c) Prohibiting misrepresentation of the origin of that material, or
    requiring that modified versions of such material be marked in
    reasonable ways as different from the original version; or

    d) Limiting the use for publicity purposes of names of licensors or
    authors of the material; or

    e) Declining to grant rights under trademark law for use of some
    trade names, trademarks, or service marks; or

    f) Requiring indemnification of licensors and authors of that
    material by anyone who conveys the material (or modified versions of
    it) with contractual assumptions of liability to the recipient, for
    any liability that these contractual assumptions directly impose on
    those licensors and authors.
```

All other non-permissive additional terms are considered ”further restrictions” within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An ”entity transaction” is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A ”contributor” is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's ”contributor version”.

A contributor's ”essential patent claims” are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, ”control” includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a ”patent license” is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To ”grant” such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a

```
publicly available network server or other readily accessible means,
then you must either (1) cause the Corresponding Source to be so
available, or (2) arrange to deprive yourself of the benefit of the
patent license for this particular work, or (3) arrange, in a manner
consistent with the requirements of this License, to extend the patent
license to downstream recipients.  "Knowingly relying" means you have
actual knowledge that, but for the patent license, your conveying the
covered work in a country, or your recipient's use of the covered work
in a country, would infringe one or more identifiable patents in that
country that you have reason to believe are valid.

  If, pursuant to or in connection with a single transaction or
arrangement, you convey, or propagate by procuring conveyance of, a
covered work, and grant a patent license to some of the parties
receiving the covered work authorizing them to use, propagate, modify
or convey a specific copy of the covered work, then the patent license
you grant is automatically extended to all recipients of the covered
work and works based on it.

  A patent license is "discriminatory" if it does not include within
the scope of its coverage, prohibits the exercise of, or is
conditioned on the non-exercise of one or more of the rights that are
specifically granted under this License.  You may not convey a covered
work if you are a party to an arrangement with a third party that is
in the business of distributing software, under which you make payment
to the third party based on the extent of your activity of conveying
the work, and under which the third party grants, to any of the
parties who would receive the covered work from you, a discriminatory
patent license (a) in connection with copies of the covered work
conveyed by you (or copies made from those copies), or (b) primarily
for and in connection with specific products or compilations that
contain the covered work, unless you entered into that arrangement,
or that patent license was granted, prior to 28 March 2007.

  Nothing in this License shall be construed as excluding or limiting
any implied license or other defenses to infringement that may
otherwise be available to you under applicable patent law.

  12. No Surrender of Others' Freedom.

  If conditions are imposed on you (whether by court order, agreement or
otherwise) that contradict the conditions of this License, they do not
excuse you from the conditions of this License.  If you cannot convey a
covered work so as to satisfy simultaneously your obligations under this
License and any other pertinent obligations, then as a consequence you may
not convey it at all.  For example, if you agree to terms that obligate you
to collect a royalty for further conveying from those to whom you convey
the Program, the only way you could satisfy both those terms and this
License would be to refrain entirely from conveying the Program.

  13. Use with the GNU Affero General Public License.

  Notwithstanding any other provision of this License, you have
permission to link or combine any covered work with a work licensed
under version 3 of the GNU Affero General Public License into a single
combined work, and to convey the resulting work.  The terms of this
License will continue to apply to the part which is the covered work,
but the special requirements of the GNU Affero General Public License,
section 13, concerning interaction through a network will apply to the
combination as such.

  14. Revised Versions of this License.

  The Free Software Foundation may publish revised and/or new versions of
the GNU General Public License from time to time.  Such new versions will
be similar in spirit to the present version, but may differ in detail to
address new problems or concerns.

  Each version is given a distinguishing version number.  If the
Program specifies that a certain numbered version of the GNU General
Public License "or any later version" applies to it, you have the
option of following the terms and conditions either of that numbered
version or of any later version published by the Free Software
Foundation.  If the Program does not specify a version number of the
GNU General Public License, you may choose any version ever published
by the Free Software Foundation.

  If the Program specifies that a proxy can decide which future
versions of the GNU General Public License can be used, that proxy's
public statement of acceptance of a version permanently authorizes you
to choose that version for the Program.

  Later license versions may give you additional or different
permissions.  However, no additional obligations are imposed on any
author or copyright holder as a result of your choosing to follow a
later version.

  15. Disclaimer of Warranty.

  THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY
APPLICABLE LAW.  EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT
HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY
OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO,
THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE.  THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM
IS WITH YOU.  SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF
ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

  16. Limitation of Liability.

  IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING
WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS
THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY
GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE
USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF
DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD
PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS),
EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF
SUCH DAMAGES.

  17. Interpretation of Sections 15 and 16.

  If the disclaimer of warranty and limitation of liability provided
above cannot be given local legal effect according to their terms,
reviewing courts shall apply local law that most closely approximates
an absolute waiver of all civil liability in connection with the
Program, unless a warranty or assumption of liability accompanies a
copy of the Program in return for a fee.

                     END OF TERMS AND CONDITIONS

            How to Apply These Terms to Your New Programs

  If you develop a new program, and you want it to be of the greatest
possible use to the public, the best way to achieve this is to make it
free software which everyone can redistribute and change under these terms.

  To do so, attach the following notices to the program.  It is safest
to attach them to the start of each source file to most effectively
state the exclusion of warranty; and each file should have at least
the "copyright" line and a pointer to where the full notice is found.

    <one line to give the program's name and a brief idea of what it does.>
    Copyright (C) <year>  <name of author>

    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program.  If not, see <http://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

  If the program does terminal interaction, make it output a short
notice like this when it starts in an interactive mode:

    <program>  Copyright (C) <year>  <name of author>
    This program comes with ABSOLUTELY NO WARRANTY; for details type ‘show w’.
    This is free software, and you are welcome to redistribute it
    under certain conditions; type ‘show c’ for details.

The hypothetical commands ‘show w’ and ‘show c’ should show the appropriate
parts of the General Public License.  Of course, your program's commands
might be different; for a GUI interface, you would use an "about box".

  You should also get your employer (if you work as a programmer) or school,
if any, to sign a "copyright disclaimer" for the program, if necessary.
For more information on this, and how to apply and follow the GNU GPL, see
<http://www.gnu.org/licenses/>.

  The GNU General Public License does not permit incorporating your program
into proprietary programs.  If your program is a subroutine library, you
may consider it more useful to permit linking proprietary applications with
the library.  If this is what you want to do, use the GNU Lesser General
```

```
Public License instead of this License.  But first, please read
<http://www.gnu.org/philosophy/why-not-lgpl.html>.
```

14 extra/Setup.hs

```haskell
import Distribution.Simple
main = defaultMain
```

15 Genetics.hs

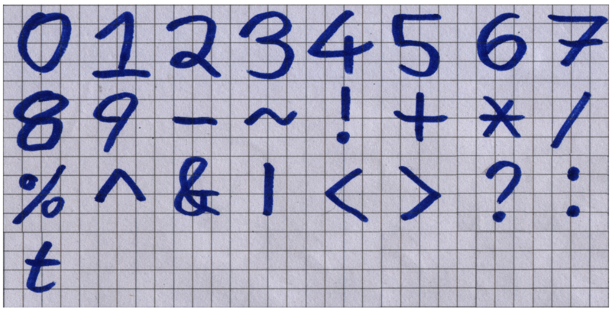
```haskell
{-# LANGUAGE ScopedTypeVariables, FlexibleInstances, Rank2Types,
    UndecidableInstances, DeriveDataTypeable #-}

-- Based on:

-- Module      :  GenProg.GenExpr.Data
-- Copyright   :  (c) 2010 Jan Snajder
-- License     :  BSD-3 (see the LICENSE file)
--
-- Maintainer  :  Jan Snajder <jan.snajder@fer.hr>
-- Stability   :  experimental
-- Portability :  non-portable
--
-- Implementation of the @GenProg.GenExpr@ interface for members of
-- the 'Data' typeclass. The implementation is based on SYB and SYZ
-- generic programming frameworks (see
-- <http://hackage.haskell.org/package/syb> and
-- <http://hackage.haskell.org/package/syz> for details).
--
-- NB: Subexpressions that are candidates for crossover points or
-- mutation must be of the same type as the expression itself, and
-- must be reachable from the root node by type-preserving traversal.
-- See below for an example.
--
------------------------------------------------------------------------------

module Genetics where

import Data.Generics
import Data.Generics.Zipper
import Data.Maybe
import Control.Monad


-- | This typeclass defines an interface to expressions
-- that can be genetically programmed.  The operations that must be
-- provided by instances of this class are used for the generation
-- of random individuals as well as crossover and mutation operations.
-- (An instance for members of the @Data@ typeclass is provided in
-- "GenProg.GenExpr.Data".)
--
-- Minimal complete definition: 'exchange', 'nodeMapM', 'nodeMapQ',
-- and 'nodeIndices'.
class GenExpr e where
  -- | Exchanges subtrees of two expressions:
  -- @exchange e1 n1 e2 n2@ replaces the subexpression of @e1@ rooted in node
  -- @n1@ with the subexpression of @e2@ rooted in @n2@, and vice versa.
  exchange :: e -> Int -> e -> Int -> (e, e)
  -- | Maps a monadic transformation function over the immediate
  -- children of the given node.
  nodeMapM :: Monad m => (e -> m e) -> e -> m e
  -- | Maps a query function over the immediate children of the given
  -- node and returns a list of results.
  nodeMapQ :: (e -> a) -> e -> [a]
  -- | A list of indices of internal (functional) and external
  -- (terminal) nodes of an expression.
  nodeIndices :: e -> ([Int], [Int])
  -- | Adjusts a subexpression rooted at the given node by applying a
  -- monadic transformation function.
  adjustM :: (Monad m) => (e -> m e) -> e -> Int -> m e
  -- | Number of nodes an expression has.
  nodes :: e -> Int
  -- | The depth of an expression. Equals 1 for single-node expressions.
  depth :: e -> Int

{-
  -- | Default method (expensive because it calls exchange twice).
  adjustM f e n = replace e n ‘liftM‘ f (get e n)
    where get e n = fst $ exchange e 0 e n
          replace e1 n1 e2 = fst $ exchange e1 n1 e2 0
-}
  nodes = (+1) . foldr (+) 0 . nodeMapQ nodes

  depth = (+1) . foldr max 0 . nodeMapQ depth


instance (Data a) => GenExpr a where

  -- | Exchanges two expression nodes. Works by using two generic
  -- zippers and exchanging their holes.
  exchange e1 n1 e2 n2 = (fromZipper y1, fromZipper y2)
    where z1 = typeMoveForUnsafe n1 $ toZipper e1
          z2 = typeMoveForUnsafe n2 $ toZipper e2
          (y1,y2) = exchangeHoles z1 z2

  -- | Adjust an expression node. Works by applying a monadic
  -- tranformation on a zipper hole.
  adjustM f e n = fromZipper ‘liftM‘ transM (mkM f) z
    where z = typeMoveForUnsafe n (toZipper e)

  nodeMapM f = gmapM (mkM f)

  nodeMapQ q (x::a) = concat $ gmapQ ([] ‘mkQ‘ (\(y::a) -> [q y])) x

  nodeIndices = index 0 [] [] . toZipper

-- Zipper moves

type Move a = Zipper a -> Maybe (Zipper a)

backtrack :: (Typeable a) => Move a
backtrack z = do
  z2 <- up z
```

```haskell
  right z2 `mplus` backtrack z2

repeatM :: (Monad m) => Int -> (a -> m a) -> a -> m a
repeatM 0 _ x = return x
repeatM n f x = f x >>= repeatM (n - 1) f

-- Moves zipper to next node in DFS order, but does not move down the
-- zipper if node satisfies query 'q'.
nextDfsQ :: Typeable a => GenericQ Bool -> Move a
nextDfsQ q z = (if query q z then Nothing else down' z)
  `mplus` right z `mplus` backtrack z

-- Moves the zipper to node 'n' from current position in DFS order,
-- skipping nodes not satisfying query 'q2' and descending only down
-- the nodes satisfying query 'q1'.
moveForQ :: (Typeable a) => GenericQ Bool -> GenericQ Bool -> Int -> Move a
moveForQ _  _  0 z = Just z
moveForQ q1 q2 n z = do
  z2 <- nextDfsQ q1 z
  moveForQ q1 q2 (if query q2 z2 then n - 1 else n) z2

-- Moves the zipper to node 'n' from current position in DFS order,
-- counting only nodes of type 'a', and not descending down the nodes
-- of other type.
typeMoveFor :: (Typeable a) => Int -> Move a
typeMoveFor n (z::Zipper a) =
  moveForQ (True `mkQ` (\(_::a) -> False)) (False `mkQ` (\(_::a) -> True)) n z

-- | Same as typeMoveFor, but throws an error if node index is out of
-- bound.
typeMoveForUnsafe :: (Typeable a) => Int -> Zipper a -> Zipper a
typeMoveForUnsafe n z = fromMaybe
  (error $ "Genetics.typeMoveForUnsafe: Nonexisting node.")
  (typeMoveFor n z)

-- | Exchanges two zipper holes.
exchangeHoles :: (Data a) => Zipper a -> Zipper a -> (Zipper a, Zipper a)
exchangeHoles (z1::Zipper a) (z2::Zipper a) = (y1,y2)
  where Just h1 = getHole z1 :: Maybe a
        Just h2 = getHole z2 :: Maybe a
        y1 = setHole h2 z1
        y2 = setHole h1 z2

index :: (Data a) => Int -> [Int] -> [Int] -> Zipper a -> ([Int], [Int])
index i is es (z :: Zipper a) =
  maybe (is2,es2) (index (i + 1) is2 es2) (typeMoveFor 1 z)
  where Just h = getHole z :: Maybe a
        (is2,es2) = if terminalQ h then (is,i:es) else (i:is,es)

terminalQ :: (Data a) => a -> Bool
terminalQ = null . nodeMapQ id

{- $Example

Suppose you have a datatype defined as

@
data E = A E E
        | B String [E]
        | C
  deriving (Eq,Show,Typeable,Data)
@

and an expression defined as

@
e = A (A C C) (B \"abc\" [C,C])
@

The subexpressions of a @e@ are considered to be only the subvalues of
@e@ that are of the same type as @e@.  Thus, the number of nodes of
expression @e@ is

>>> nodes e
5

because subvalues of node @B@ are of different type than expression
@e@ and therefore not considered as subexpressions.

Consequently, during a genetic programming run, subexpressions that
are of a different type than the expression itself, or subexpression
that cannot be reached from the root node by a type-preserving
traversal, cannot be chosen as crossover points nor can they be
mutated.

-}
```

16 .gitignore

```
*.hi
*.o
bitbreeder
bitbreeder_video
bitbreeder_audio
bitbreeder_judge
glyphs.ppm
glyphs.raw
a
v
o
.cabal-sandbox
cabal.sandbox.config
dist
dist-newstyle
```

17 glyphs.png



18 go.c

```
#include <limits.h>

static inline int safe_div(int a, int b) { if ((b == 0) || (a == INT_MIN && b == -1)) { return 0; } else { return a / b; } }
static inline int safe_mod(int a, int b) { if ((b == 0) || (a == INT_MIN && b == -1)) { return 0; } else { return a % b; } }

const char code[] = CODE;
float F(int t) { return (((T)&255)-128)/256.0f; }

typedef struct {
  int t;
  int k;
  float a[4];
  float dc;
} S;

float go(S *s) {
  s->k += 1;
  if (s->k == 6) {
    s->k = 0;
    s->t += 1;
    s->a[0] = s->a[1];
    s->a[1] = s->a[2];
    s->a[2] = s->a[3];
    s->a[3] = F(s->t);
  }
  /*
    -- http://en.wikipedia.org/wiki/Cubic_Hermite_spline#Interpolation_on_the_unit_interval_without_exact_derivatives
    putStr . unlines . map (show . map (/2)) $ [
      [ -x^3 + 2*x^2 - x
      , 3 * x^3 - 5 * x^2 + 2
      , -3 * x^3 + 4 * x^2 + x
      , x^3 - x^2
      ] | i <- [0..5], let x = i / 6 ]
  */
  const float c[6][4] =
    { {0.0,1.0,0.0,0.0}
    , {-5.7870370370370364e-2,0.9375,0.13194444444444442,-1.1574074074074073e-2}
    , {-7.407407407407407e-2,0.7777777777777778,0.3333333333333333,-3.7037037037037035e-2}
    , {-6.25e-2,0.5625,0.5625,-6.25e-2}
    , {-3.7037037037037035e-2,0.33333333333333326,0.7777777777777777,-7.407407407407407e-2}
    , {-1.157407407407407e-2,0.13194444444444442,0.9375,-5.787037037037035e-2}
    };
  float a = c[s->k][0] * s->a[0] + c[s->k][1] * s->a[1] + c[s->k][2] * s->a[2] + c[s->k][3] * s->a[3];
  s->dc = s->dc * 0.99 + 0.01 * a;
  return a - s->dc;
}
```

19 gradient.ppm

20 judge.c

```
#define _POSIX_C_SOURCE 1
#include <signal.h>

#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <limits.h>
#include <stdlib.h>

#ifdef JUDGE_SOUNDFILE
#include <sndfile.h>
#else
#include <dlfcn.h>
#endif

#include <fftw3.h>
#define wisdomfile "/run/shm/bitbreeder.fftw"

static const double pi = 3.141592653589793;
static const double sr = 8192.0;
#define blocksize 2048
#define overlap 4
#define BINS (blocksize / 2 + 1)
#define PARAMS 6
#define LEVELS 11

#define N (((1 << LEVELS) + overlap - 1) * blocksize / overlap)

struct audio {
  int length;
  float *data;
};

typedef float F_t(int);
```

```
struct audio *audio(const char *name) {
  struct audio *a = calloc(1, sizeof(*a));
  a->length = N;
  a->data = calloc(a->length, sizeof(*a->data));
#ifdef JUDGE_SOUNDFILE
  SF_INFO info; memset(&info, 0, sizeof(info));
  SNDFILE *in = sf_open(name, SFM_READ, &info);
  sf_readf_float(in, a->data, a->length);
  sf_close(in);
#else
  void *dl = dlopen(name, RTLD_NOW);
  F_t *cb;
  *(void **) (&cb) = dlsym(dl, ”F”);
  float dc = cb(0);
  for (int t = 0; t < a->length; ++t) {
    float x = cb(t);
    dc = dc * 0.99 + 0.01 * x;
    a->data[t] = x - dc;
  }
  dlclose(dl);
#endif
  return a;
}

struct frame {
  float loudness;
  float spectrum[BINS];
};

struct frames {
  int current;
  struct frame *frame;
};

struct frames *frames(struct audio *a) {
  struct frames *f = calloc(1, sizeof(*f));
  f->current = 0;
  f->frame = calloc(1 << LEVELS, sizeof(*f->frame));
  float *ibuf = fftwf_alloc_real(blocksize);
  float *obuf = fftwf_alloc_real(blocksize);
  fftwf_import_wisdom_from_filename(wisdomfile);
  fftwf_plan plan = fftwf_plan_r2r_1d(blocksize, ibuf, obuf, FFTW_R2HC, ↙
      ↳ FFTW_DESTROY_INPUT | FFTW_EXHAUSTIVE);
  fftwf_export_wisdom_to_filename(wisdomfile);
  float *window = calloc(blocksize, sizeof(*window));
  for (int t = 0; t < blocksize; ++t) {
    window[t] = 0.5 - 0.5 * cos(t * 2 * pi / blocksize);
  }
  float rsqrtblocksize = 1 / sqrtf(blocksize);
  for (int i = 0; i < 1 << LEVELS; ++i) {
    int b = i * blocksize / overlap;
    double l = 0;
    for (int t = 0; t < blocksize; ++t) {
      float x = a->data[b + t];
      ibuf[t] = window[t] * x;
      l += window[t] * x * x;
    }
    f->frame[i].loudness = sqrt(l) * rsqrtblocksize;
    fftwf_execute(plan);
    f->frame[i].spectrum[0     ] = fabsf(obuf[0]) * rsqrtblocksize;
    f->frame[i].spectrum[BINS-1] = fabsf(obuf[blocksize/2]) * rsqrtblocksize;
    for (int k = 1; k < BINS-1; ++k) {
      float re = obuf[k];
      float im = obuf[blocksize - k];
      f->frame[i].spectrum[k] = sqrtf(re * re + im * im) * rsqrtblocksize;
    }
  }
  return f;
}

enum {
  p_loudness = 0,
  p_tonality,       // min(1, (log10 m1 - 10 sum (log10 a_k) / sum 1)/60)
  p_centroid,       // m1 = sum a_k f_k / sum a_k
  p_deviation,      // m2 = sum (f_k - m1)^2 a_k / sum a_k; s^2 = m2
  p_skewness,       // m3 = sum (f_k - m1)^3 a_k / sum a_k; g1 = m3 / s^3
  p_kurtosis,       // m4 = sum (f_k - m1)^4 a_k / sum a_k; g2 = m4 / s^4
};

struct statistic {
  double s0, s1, s2;
};

void statistic(double x, struct statistic *s) {
  s->s0 = 1;
  s->s1 = x;
  s->s2 = x*x;
}

void wstatistic(double w, double x, struct statistic *s) {
  if (isnan(w)) { w = 0; }
  if (isnan(x)) { x = 0; }
  s->s0 = w;
  s->s1 = w*x;
  s->s2 = w*x*x;
}

void combine(struct statistic *x, struct statistic *y, struct statistic *r) {
  r->s0 = x->s0 + y->s0;
  r->s1 = x->s1 + y->s1;
  r->s2 = x->s2 + y->s2;
}

double mean(struct statistic *s) {
  if (! isnan(s->s1) && s->s0 > 0) {
    return s->s1 / s->s0;
  }
  return 0;
}

double stddev(struct statistic *s) {
  double d = sqrt(s->s0 * s->s2 - s->s1 * s->s1) / s->s0;
  if (! (d >= 0)) { d = 0; }
  return d;
```

```
}

struct analysis {
  struct statistic base;
  struct statistic levels[LEVELS+1];
};

void combines(int depth, struct analysis *x, struct analysis *y, struct analysis *r) {
  combine(&x->base, &y->base, &r->base);
  for (int level = 0; level < depth; ++level) {
    combine(&x->levels[level], &y->levels[level], &r->levels[level]);
  }
  statistic(stddev(&r->base), &r->levels[depth]);
}

struct analyses {
  struct analysis param[PARAMS];
};

void analyse(struct frames *f, struct analyses *a) {
  struct frame *s = &f->frame[f->current++];
  statistic(s->loudness, &a->param[p_loudness].base);
  {
    double s0 = 0, s1 = 0, s2 = 0, s3 = 0, s4 = 0;
    for (int k = 0; k < BINS; ++k) {
      double x = s->spectrum[k];
      double f = k * 2.0 / blocksize;
      s0 += x;
      s1 += x * f;
    }
    if (! (s0 > 0)) { s0 = 1; }
    double m1 = s1 / s0;
    for (int k = 0; k < BINS; ++k) {
      double x = s->spectrum[k];
      double f = k * 2.0 / blocksize;
      double y = f - m1;
      s2 += x * y * y;
      s3 += x * y * y * y;
      s4 += x * y * y * y * y;
    }
    double m2 = s2 / s0;
    double m3 = s3 / s0;
    double m4 = s4 / s0;
/*
    statistic(m1,                &a->param[p_centroid ].base);
    statistic(sqrt(m2),          &a->param[p_deviation].base);
    statistic(m3 / pow(m2, 1.5), &a->param[p_skewness ].base);
    statistic(m4 / pow(m2, 2),   &a->param[p_kurtosis ].base);
*/
    double l = s->loudness;
    wstatistic(l, m1,                &a->param[p_centroid ].base);
    wstatistic(l, sqrt(m2),          &a->param[p_deviation].base);
    wstatistic(l, m3 / pow(m2, 1.5), &a->param[p_skewness ].base);
    wstatistic(l, m4 / pow(m2, 2),   &a->param[p_kurtosis ].base);
  }
  {
    double s0 = 0, s1 = 0, sL = 0;
    for (int k = 0; k < BINS; ++k) {
      double x = s->spectrum[k] + 1e-6;
      s0 += 1;
      s1 += x * x;
      sL += log(x * x);
    }
    double tonality = 1 - exp(sL / s0) / (s1 / s0);
    if (! (0 < tonality)) { tonality = 0; }
    if (! (1 > tonality)) { tonality = 1; }
/*
    statistic(tonality, &a->param[p_tonality].base);
*/
    wstatistic(s->loudness, tonality, &a->param[p_tonality].base);
  }
}

void combiness(int depth, struct analyses *x, struct analyses *y, struct analyses *r) {
  for (int param = 0; param < PARAMS; ++param) {
    combines(depth, &x->param[param], &y->param[param], &r->param[param]);
  }
}

void recurse(struct frames *f, int depth, struct analyses *a) {
  if (depth) {
    struct analyses x, y;
    recurse(f, depth - 1, &x);
    recurse(f, depth - 1, &y);
    combiness(depth - 1, &x, &y, a);
  } else {
    analyse(f, a);
  }
}

struct result {
  double average, variability, granularity;
};

struct results {
  struct result result[PARAMS];
  double novelty;
};

void judge(struct frames *f, struct results *r) {
  struct analyses a;
  recurse(f, LEVELS, &a);
  for (int param = 0; param < PARAMS; ++param) {
    r->result[param].average     = mean(&a.param[param].base);
    r->result[param].variability = mean(&a.param[param].levels[0]);
    struct statistic s, t;
    wstatistic(0, 0, &t);
    for (int level = 1; level < LEVELS; ++level) {
      wstatistic(level, mean(&a.param[param].levels[level]), &s);
      combine(&s, &t, &t);
    }
    r->result[param].granularity = mean(&t);
  }
}

const char *name = 0;

void fpe_handler(int s) {
  (void) s;
  fprintf(stderr, "SIGFPE %s\n", name);
  exit(1);
}

int main(int argc, char **argv) {
  (void) argc;
  name = argv[1];
  struct sigaction act, old;
  act.sa_handler = fpe_handler;
  sigemptyset(&act.sa_mask);
  sigaction(SIGFPE, &act, &old);
  struct results r;
  judge(frames(audio(argv[1])), &r);
  r.novelty = atoi(argv[2]);
  fwrite(&r, sizeof(r), 1, stdout);
  sigaction(SIGFPE, &old, 0);
  return 0;
}
```

21 live.c

```
#define _GNU_SOURCE
#include <stdio.h>
#include <stdlib.h>

#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>

#include <dlfcn.h>

#include <jack/jack.h>

// per-sample callback implemented in go.so
typedef float callback(void *);

// default silent callback
static float deffunc(void *data) {
  return 0;
}

static struct {
  jack_client_t *client;
  jack_port_t *out;
  void *data;
  callback * volatile func;
} state;

// race mitigation
volatile int inprocesscb = 0;
```

```

static int processcb(jack_nframes_t nframes, void *arg) {
  inprocesscb = 1; // race mitigation
  jack_default_audio_sample_t *out = (jack_default_audio_sample_t *) ↙
      ↳ jack_port_get_buffer(state.out, nframes);
  callback *f = state.func;
  for (jack_nframes_t i = 0; i < nframes; ++i) {
    out[i] = f(state.data);
  }
  inprocesscb = 0; // race mitigation
  return 0;
}

static void errorcb(const char *desc) {
  fprintf(stderr, "JACK error: %s\n", desc);
}

static void shutdowncb(void *arg) {
  exit(1);
}

static void atexitcb(void) {
  jack_client_close(state.client);
}

int main(int argc, char **argv) {
  srand(time(0));
  state.func = deffunc;
  state.data = calloc(1, 1024 * 1024);
  jack_set_error_function(errorcb);
  if (!(state.client = jack_client_open("live", JackNoStartServer, 0))) {
    fprintf(stderr, "jack server not running?\n");
    return 1;
  }
  atexit(atexitcb);
  jack_set_process_callback(state.client, processcb, 0);
  jack_on_shutdown(state.client, shutdowncb, 0);
  // mono processing
  state.out = jack_port_register(state.client, "output_1", ↙
      ↳ JACK_DEFAULT_AUDIO_TYPE, JackPortIsOutput, 0);
  if (jack_activate(state.client)) {
    fprintf (stderr, "cannot activate JACK client");
    return 1;
  }
  // mono recording
  if (jack_connect(state.client, "live:output_1", "record:in_1")) {
    fprintf(stderr, "cannot connect to recorder\n");
  }
  // stereo output
  const char **ports;
  if ((ports = jack_get_ports(state.client, NULL, NULL, JackPortIsPhysical | ↙
      ↳ JackPortIsInput))) {
    int i = 0;
    while (ports[i] && i < 2) {
      if (jack_connect(state.client, jack_port_name(state.out), ports[i])) {
        fprintf(stderr, "cannot connect output port\n");
      }
```

```
        i++;
      }
      free(ports);
    }
    void *old_dl = 0;
    void *new_dl = 0;
    char *soname = 0;
    while (1 == scanf("%ms", &soname)) {
      if ((new_dl = dlopen(soname, RTLD_NOW))) {
        callback *new_cb;
        *(void **) (&new_cb) = dlsym(new_dl, "go");
        if (new_cb) {
          // race mitigation: dlclose with jack running in .so -> boom
          while (inprocesscb) ;
          state.func = new_cb;
          if (old_dl) {
            dlclose(old_dl);
          }
          old_dl = new_dl;
        } else {
          dlclose(new_dl);
          new_dl = 0;
        }
      } else {
      }
      free(soname);
      soname = 0;
    }
    return 0;
  }
```

22 Makefile

```
all: bitbreeder bitbreeder_video bitbreeder_audio bitbreeder_judge glyphs.raw

clean:
	rm -f bitbreeder bitbreeder.o bitbreeder_video bitbreeder_video.o bitbreeder_video.hi bitbreeder_audio bitbreeder_judge bitbreeder.hi Database.o Database.hi Population.o Population.hi Expression.hi Expression.o Evolve.hi Evolve.o Compile.hi Compile.o Video.hi Video.o Genetics.o Genetics.hi Metric.o Metric.hi Config.o Config.hi judge_sf debug glyphs.raw glyphs.ppm

bitbreeder: bitbreeder.hs Database.hs Population.hs Expression.hs Evolve.hs Compile.hs Genetics.hs Metric.hs Config.hs .cabal-sandbox
	cabal v1-exec -- ghc -Wall -threaded -O2 bitbreeder.hs

bitbreeder_video: bitbreeder_video.hs Expression.hs Video.hs Config.hs .cabal-sandbox
	cabal v1-exec -- ghc -Wall -threaded -O2 bitbreeder_video.hs

bitbreeder_audio: live.c
	gcc -std=c99 -Wall -pedantic -Wextra -Wno-unused-parameter -O3 -o bitbreeder_audio live.c `pkg-config --cflags --libs jack` -ldl -lm

bitbreeder_judge: judge.c
	gcc -std=c99 -Wall -pedantic -Wextra -O3 -ggdb -o bitbreeder_judge judge
```

```
            ↳ .c -lm -lfftw3f -ldl

judge_sf: judge.c
        gcc -std=c99 -Wall -pedantic -Wextra -O3 -ggdb -o judge_sf judge.c -lm -↙
            ↳ lfftw3f -lsndfile -DJUDGE_SOUNDFILE

debug: debug.c
        gcc -std=c99 -Wall -pedantic -Wextra -Wno-unused-parameter -O3 -o debug ↙
            ↳ debug.c -ldl

glyphs.raw: glyphs.png
        convert glyphs.png glyphs.ppm && tail -c 1572864 < glyphs.ppm > glyphs.↙
            ↳ raw

.cabal-sandbox:
        cabal v1-sandbox init
        cabal v1-install cairo MonadRandom GLFW-b gtk OpenGLRaw process stm syb ↙
            ↳ syz time vector --reorder-goals
```

23 Metric.hs

```
{-# LANGUAGE ForeignFunctionInterface #-}
module Metric where

import Prelude hiding (read, replicate, sum, zip, zip3, zipWith, zipWith3)

import Control.Monad (when)
import Data.Vector.Unboxed
import qualified Data.Vector.Storable as S (thaw, unsafeFreeze)
import qualified Data.Vector.Storable.Mutable as S (unsafeWith)

import Foreign (allocaBytes, copyBytes)
import System.IO (Handle, hGetBuf)
import System.IO.Error (mkIOError, eofErrorType)

newtype Metric   = M (Vector (Double, Double))
newtype Stats    = S (Vector (Double, Double, Double))
newtype Analysis = A (Vector Double)
newtype Weight   = W (Vector Double)
newtype Target   = T (Vector Double)
type    Score    = Double

emptyMetric    :: Metric
emptyMetric    =  M $ zip zero zero

emptyStats     :: Stats
emptyStats     =  S $ zip3 zero zero zero

emptyAnalysis  :: Analysis
emptyAnalysis  =  A zero

emptyWeight    :: Weight
emptyWeight    =  W zero

emptyTarget    :: Target
emptyTarget    =  T zero
```

```
zero :: Vector Double
zero = replicate elements 0

insert :: Analysis -> Stats -> Stats
insert (A as) (S ss) = S $ zipWith f as ss
  where
    f a (s0, s1, s2) = (s0 + 1, s1 + a, s2 + a * a)

delete :: Analysis -> Stats -> Stats
delete (A as) (S ss) = S $ zipWith f as ss
  where
    f a (s0, s1, s2) = (s0 - 1, s1 - a, s2 - a * a)

target :: Target -> Weight -> Stats -> Metric
target (T ts) (W ws) (S ss) = M $ zipWith3 f ws ts ss
  where
    f w t (s0, s1, s2)
      | isInfinite tt = (0, 0)
      | isNaN      tt = (0, 0)
      | isInfinite ww = (0, 0)
      | isNaN      ww = (0, 0)
      | otherwise     = (tt, ww)
      where
        mean = s1 / s0
        stddev = sqrt (s0 * s2 - s1 * s1) / s0
        tt = mean + stddev * t
        ww = w / stddev

score :: Metric -> Analysis -> Score
score (M ms) = \(A as) -> sum $ zipWith f ms as
  where
    f (t, w) = \a -> let d = w * (a - t) in d * d

read :: Handle -> IO Analysis
read h = allocaBytes bytes $ \ptr -> do
  bytes' <- hGetBuf h ptr bytes
  when (bytes /= bytes') $ ioError (mkIOError eofErrorType "" Nothing Nothing)
  m <- S.thaw (convert zero)
  S.unsafeWith m $ \q -> copyBytes q ptr bytes
  (A . convert) ‘fmap‘ S.unsafeFreeze m

bytes, elements, measurements, parameters :: Int
bytes = elements * 8
elements = measurements * parameters + 1
measurements = 6
parameters = 3
```

24 Population.hs

```
module Population(Item(..), Population, empty, update, target, toAscList) where

import Database (DB)
import qualified Database as D
import Metric (Stats, Analysis, Target, Weight)
import qualified Metric as S
import Expression (E())
```

```
data Item = Item{ itemID :: !Int, itemExpr :: E, itemMetric :: Analysis }

type Population = (DB Item, Stats, Target, Weight)

empty :: Population
empty = (D.empty, S.emptyStats, S.emptyTarget, S.emptyWeight)

insert :: Item -> Population -> Population
insert it (d, s, t, w) = (D.insert it d, S.insert (itemMetric it) s, t, w)

toAscList :: Population -> [Item]
toAscList (d, _, _, _) = D.toAscList d

prune :: Int -> Population -> Population
prune n p@(d, s, t, w) = case fmap D.toAscList $ D.splitAt n d of
  (_, []) -> p
  (keep, discard) -> (keep, foldr S.delete s $ map itemMetric discard, t, w)
{-
   foldr S.insert S.emptyStats . map itemMetric . D.toAscList $ k, t, w)
  let (keep, discard) = D.splitAt n d
  in  (keep,  . D.toAscList $ discard, t, w)
-}

sort :: Population -> Population
sort (d, s, t, w) = (D.sortOn (S.score (S.target t w s) . itemMetric) d, s, t, w)

update :: Item -> Population -> Population
update it = prune maxPopCount . insert it

target :: Target -> Weight -> Population -> Population
target t w (d, s, _, _) = sort (d, s, t, w)

maxPopCount :: Int
maxPopCount = 1024
```

25 README

```
BitBreeder
==========


BitBreeder evolves noisy expressions.


Build Requirements
------------------

BitBreeder is written in Haskell and C.  You need GHC (tested with ghc-7.6.2)
and (at least) these libraries from Hackage (or elsewhere):

  cairo gtk gtkglext MonadRandom OpenGLRaw process stm syb syz time vector

You need GCC (tested with gcc-4.7.2) and (at least) these libraries:

  m dl jack fftw3f

You need ImageMagick for PNG to PPM conversion.

You need Make to build it all.


Runtime Requirements
--------------------

You need JACK (running at 48000Hz), GCC (BitBreeder generates source code and
compiles it) and ecasound (for recording).  After running you can encode a
video, which needs recent versions of avprobe and avconv (tested with 1.0.5
for Debian Wheezy from deb-multimedia repository, older versions like stock
Wheezy will cause problems).


Usage
-----

qjackctl &   # set up JACK and start jackd; stop the transport and rewind it
for CPU in 0 1 2 3 ... ; do sudo cpufreq-set -c ${CPU} -g performance ; done
./start.sh
./encode.sh a/SESSION
mplayer a/SESSION.mkv

The main BitBreeder window has a bunch of sliders.  Each row corresponds to
an audio descriptor, the left slider is the normalized target value and the
right slider is the weighting.  The other window displays the currently
sounding expression, which is the fittest expression matching the current
tab.  You can create more tabs, each with their own fitness target/weights.
BitBreeder cross-breeds and mutates the expression populations from each tab.

Targets are normalized: the center of the slider range is the mean of the
population, and the extremes of the slider range are +/- a few standard
deviations of the population.   Weighting is from 0 at the left increasing
to the right, changing a target will have no effect when its weight is 0.
When all weights are zero each newly generated expression is deemed the
fittest.


Implementation
--------------

BitBreeder consists of a few programs: user interface and control logic
(bitbreeder), the visualisation of the expression (bitbreeder_video),
live audio generation (bitbreeder_audio), and expression audio analysis
(bitbreeder_judge).


bitbreeder
----------

FIXME cross-breeding, mutation, populations, compilation


bitbreeder_judge
----------------

The judge loads an .so containing the compiled expression, and generates
```

a couple of minutes of audio with it. The main loop is a tree structure:

```
    recursion depth
    :    /
    2   /\
       /  \
    1 /\  /\
    0/\/\/\/\/
     01234567.. audio frames
```

Each audio frame is an FFT spectrum and RMS volume for that block (which are windowed and overlapped).

At the lowest level of the tree the basic instantaneous descriptors are calculated (tonality, spectral centroid, etc...), each stored as a statistic (weight, weight * value, weight * value^2) with the weight usually based on the RMS volume.

Each subsequent level combines statistics from all the previous levels - each node (for each basic descriptor) combines pairwise two lists of statistics (ordered by level) and adds a new statistic as the mean of the level below it.

Example (assuming weight is 1):

```
-> input descriptor sequence
  7
  4
  5
  1
-> base level
  1 7 49
  1 4 16
  1 5 25
  1 1  1
-> next level
  2 11 98     1 5.5 30.25
  2  6 36     1 3     9
-> top level
  4 17 134    2 8    39.25     1 4 16
-> final results
  mean(0) = 17/4 = 4.25
  stddev(0) = sqrt(4 * 134 - 17^2) / 4 = 3.929
  stddev(1) = sqrt(2 * 39.25 - 8^2) / 2 = 1.936
  stddev(2) = sqrt(1 * 16 - 4^2) / 1 = 0
  granularity = centroid of stddev(level)
    = (0 * 3.929 + 1 * 1.936 + 2 * 0) / (0 + 1 + 2)
    = 0.64549
-> output
  mean(0), stddev(0), granularity
```

bitbreeder_audio

The main loop watches stdin for names of an .so containing a compiled expression. For each line, it loads the .so and swaps the JACK process

```
callback to the new expression (taking care not to unload code that is
currently running).

FIXME sample rate conversion, DC offset removal


bitbreeder_video
----------------

The main loop watches stdin for expressions in Haskell's Show syntax.
For each line, it parses the expression, lays it out as a tree, and
displays it.

FIXME glyph map, textureQueryLod


Legal
-----

Copyright (C) 2013  Claude Heiland-Allen <claude@mathr.co.uk>
License: GPLv3+
Warranty: NONE


--
https://mathr.co.uk
```

26 spectrogram.c

```c
#include <math.h>
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
#include <string.h>
#include <sndfile.h>
#include <fftw3.h>

#define wisdomfile "/run/shm/bitbreeder.fftw"

static const double pi = 3.141592653589793;
static const double sr = 8192.0;
static const int bs = 2048;
static const int ol = 4;

struct audio {
  int channels;
  int frames;
  float *data;
};

static struct audio *audio_load(const char *filename) {
  struct audio *audio = calloc(1, sizeof(*audio));
  if (! audio) {
    return 0;
  }
  SF_INFO info; memset(&info, 0, sizeof(info));
  SNDFILE *in = sf_open(filename, SFM_READ, &info);
  if (! in) {
    free(audio);
    return 0;
  }
  audio->channels = info.channels;
  audio->frames = info.frames;
  audio->data = calloc(audio->channels * audio->frames, sizeof(*audio->data));
  if (! audio->data) {
    sf_close(in);
    free(audio);
    return 0;
  }
  sf_readf_float(in, audio->data, audio->frames);
  sf_close(in);
  return audio;
}

struct fft {
  float *window;
  float *in;
  float *out;
  float *ibuf;
  float *obuf;
  fftwf_plan plan;
};

static struct fft *fft_init() {
  struct fft *fft = calloc(1, sizeof(*fft));
  if (! fft) {
    return 0;
  }
  fft->window = calloc(bs, sizeof(*fft->window));
  fft->in = calloc(bs, sizeof(*fft->in));
  fft->out = calloc(bs, sizeof(*fft->out));
  fft->ibuf = fftwf_alloc_real(bs);
  fft->obuf = fftwf_alloc_real(bs);
  for (int t = 0; t < bs; ++t) {
    fft->window[t] = 0.5 - 0.5 * cos(t * 2 * pi / bs);
  }
  fftwf_import_wisdom_from_filename(wisdomfile);
  fft->plan = fftwf_plan_r2r_1d(bs, fft->ibuf, fft->obuf, FFTW_R2HC, ↙
      ↳ FFTW_DESTROY_INPUT | FFTW_EXHAUSTIVE);
  fftwf_export_wisdom_to_filename(wisdomfile);
  return fft;
}

static void fft_compute(struct fft *fft) {
  for (int k = 0; k < bs; ++k) {
    fft->ibuf[k] = fft->window[k] * fft->in[k];
  }
  fftwf_execute(fft->plan);
  fft->out[0] = fft->obuf[0] * fft->obuf[0] / sqrtf(bs);
  fft->out[bs/2] = fft->obuf[bs/2] * fft->obuf[bs/2] / sqrtf(bs);
  for (int k = 1; k < bs/2; ++k) {
    float re = fft->obuf[k];
    float im = fft->obuf[bs - k];
    fft->out[k] = sqrtf(re * re + im * im) / sqrtf(bs);
  }
}

struct plane {
  int width;
  int height;
  float *data;
};

struct planes {
  int count;
  struct plane *plane;
};

static void planes_copy(float *src, int count, struct planes *planes, int p, int x) {
  for (int i = 0; i < count; ++i) {
    planes->plane[p].data[planes->plane[p].width * i + x] = src[i];
  }
}

static void audio_copy(float *dst, int count, struct audio *audio, int t0, int c) {
  int t = t0;
  for (int i = 0; i < count; ++i) {
    if (0 <= t && t < audio->frames) {
      dst[i] = audio->data[t * audio->channels + c];
    } else {
      dst[i] = 0;
    }
    ++t;
  }
}

static struct planes *planes_init(struct audio *audio, struct fft *fft, int count) {
  struct planes *planes = calloc(1, sizeof(*planes)); // FIXME cleanup
  planes->count = count;
  planes->plane = calloc(planes->count, sizeof(*planes->plane)); // FIXME cleanup
  int i = 0;
//  for (int i = 0; i < planes->count; ++i) {
//    fprintf(stderr, ”%d\n”, i);
    double frames = 2 * bs + audio->frames;
    planes->plane[i].width = ceil(frames * ol / bs);
    planes->plane[i].height = bs;
    planes->plane[i].data = calloc(planes->plane[i].width * planes->plane[i].height, sizeof(*planes->plane[i].data)); // FIXME cleanup
    int x = 0;
    for (int t = -bs; t < bs + audio->frames; t += bs / ol) {
      for (int c = 0; c < audio->channels; ++c) {
        audio_copy(fft->in, bs, audio, t, c);
        fft_compute(fft);
        planes_copy(fft->out, bs, planes, i, x);
      }
      ++x;
    }
```

```
//     struct audio *audio2 = audio_downsample(audio);
//     free(audio);
//     audio = audio2;
//   }
  return planes;
}

struct image {
  int width;
  int height;
  unsigned char *data;
};

static struct image *image_init(int frames) {
  struct image *image = calloc(1, sizeof(*image));
  if (! image) {
    return 0;
  }
  image->height = 128;
  double dframes = frames;
  image->width = ceil(dframes * ol / bs);
  image->data = calloc(1, image->width * image->height);
  if (! image->data) {
    free(image);
    return 0;
  }
  return image;
}

static void image_write(struct image *image, const char *filename) {
  FILE *out = fopen(filename, "wb");
  if (out) {
    fprintf(out, "P5\n%d %d\n255\n", image->width, image->height);
    fwrite(image->data, image->width * image->height, 1, out);
    fclose(out);
  }
}

static double planes_lookup(struct planes *planes, double f0, double f1, double t0) {
  double y0 = f0 / sr * planes->plane[0].height;
  double y1 = f1 / sr * planes->plane[0].height;
  int v0 = floor(y0);
  int v1 = ceil(y1);
  double z = 0;
  double t = t0;
  double x = ol * t / bs;
  int u = floor(x);
  int k = 0;
  for (int v = v0; v < v1; ++v) {
    if (0 <= u && u < planes->plane[0].width && 0 <= v && v < planes->plane[0].height) {
      z += planes->plane[0].data[planes->plane[0].width * v + u];
      k += 1;
    }
  }
  if (k == 0) { k = 1; }
  return z / k;
}

static void compute(struct planes *planes, struct image *image) {
  unsigned char *data = image->data;
  for (int y = 0; y < image->height; ++y) {
    double f0 = (pow(2, (y     - image->height) * 1.0 / image->height) - 0.5) * sr;
    double f1 = (pow(2, (y + 1 - image->height) * 1.0 / image->height) - 0.5) * sr;
    for (int x = 0; x < image->width; ++x) {
      double t = (x * bs) / ol + (ol - 1) * bs/ol;
      double v = planes_lookup(planes, f0, f1, t);
      unsigned char g = fmin(fmax(255 * v, 0), 255);
      *data++ = g;
    }
  }
}

int main(int argc, char **argv) {
  if (argc < 3) {
    return 1;
  }
  int retval = 1;
  struct audio *audio = audio_load(argv[1]);
  if (audio) {
    struct fft *fft = fft_init();
    if (fft) {
      int frames = audio->frames;
      struct planes *planes = planes_init(audio, fft, 1);
      if (planes) {
        struct image *image = image_init(frames);
        if (image) {
          compute(planes, image);
          image_write(image, argv[2]);
          retval = 0;
//          image_free(image);
        }
//        planes_free(planes);
      }
//      fft_free(fft);
    }
  }
  return retval;
}
```

27 start.sh

```
#!/bin/bash
# sudo cpufreq-set -c 0 -g performance
# sudo cpufreq-set -c 1 -g performance
ulimit -s unlimited
rm -rf o
rm -f v
make
SESSION="bitbreeder-$(date -u +%F-%H%M%S)"
mkdir -p o a "a/${SESSION}"
```

```
ln -s "a/${SESSION}/" v
ecasound -q -G:jack,record -f:f32,1,48000 -i:jack -o "a/${SESSION}.wav" &
sleep 5
time ./bitbreeder +RTS -N 2>"a/${SESSION}.err" >"a/${SESSION}.out"
sleep 5
kill %ecasound
echo "./encode.sh \"a/${SESSION}\""
```

28 Statistics.hs

```
type R = Double

data Stat = Stat !R !R !R

Stat s0 s1 s2 <> Stat t0 t1 t2 = Stat (s0 + t0) (s1 + t1) (s2 + t2)

stat :: R -> Stat
stat x = Stat 1 x (x*x)

mean :: Stat -> R
mean (Stat s0 s1 _) = s1 / s0

stddev :: Stat -> R
stddev (Stat s0 s1 s2)
  | t > 0 = t
  | otherwise = 0
  where t = sqrt (s0 * s2 - s1 * s1) / s0

type Stats = (Stat, [Stat])

merge :: Stats -> Stats -> Stats
merge (s, ss) (t, ts) = (r, stat (mean r) : zipWith (<>) ss ts)
  where r = s <> t

pairwise f (a:b:cs) = f a b : pairwise f cs
pairwise _ _ = [ ]

granularity :: [R] -> R
granularity
  = sum . zipWith (*) [0..] . map stddev
  . reverse . snd . head . last . takeWhile (not . null)
  . iterate (pairwise merge) . map (\x -> (stat x, []))

go :: (R -> R) -> IO ()
go f = print $ granularity [ f t | t <- [1 .. 2^16] ]

main :: IO ()
main = do
  go (\t -> sin (t / 10))
  go (\t -> sin (t / 1000))
  go (\t -> sin (t / 10) + 10 * sin (t / 1000))
  go (\t -> sin (10 / (t + 1)))

{-
6.305600061502868
43.442727416334165
434.64767794388223
0.5956135757335486
-}
```

29 stroke.frag

```
void main() {
  gl_FragColor = vec4(1.0, 0.7, 0.7, 1.0);
}
```

30 Video.hs

```
module Video (setupGL, draw, pngFilename, captureToPNG) where

import Data.Maybe (mapMaybe)
import Data.List (intercalate, transpose)
import Foreign (allocaBytes, nullPtr, peek, peekArray, plusPtr, with, withArray)
import Foreign.C (peekCStringLen, withCString)
import System.IO (hPutStrLn, stderr, hGetBuf, withBinaryFile, IOMode(ReadMode))
import Graphics.Rendering.Cairo (Format(FormatRGB24), formatStrideForWidth, ↙
    ↳ surfaceWriteToPNG, withImageSurfaceForData)
import Graphics.GL

import Config (videoW, videoH)
import Expression

type Glyph  = Char

data Layout = Layout [Glyph] (Int, Int) [Layout]

layout :: E -> Layout
layout X = Layout ”t” (1, 1) []
layout (I i) = Layout s (length s, 1) [] where s = show i
layout (U u e) = Layout s (w ‘max‘ length s, h + 1) [l]
  where
    s = glyphsU u
    l@(Layout _ (w, h) _) = layout e
layout (B b e f) = Layout s (ew + fw + 1, (eh ‘max‘ fh) + 1) [el, fl]
  where
    s = glyphsB b
    el@(Layout _ (ew, eh) _) = layout e
    fl@(Layout _ (fw, fh) _) = layout f
layout (T e f g) = Layout ”?:” (ew + fw + gw + 2, (eh ‘max‘ fh ‘max‘ gh) + 1) [↙
    ↳ el, fl, gl]
  where
    el@(Layout _ (ew, eh) _) = layout e
    fl@(Layout _ (fw, fh) _) = layout f
    gl@(Layout _ (gw, gh) _) = layout g

type Position = (Float, Float)
type Edge = (Position, Position)

layoutEdges :: Layout -> (Position, [Edge])
layoutEdges (Layout _ (w, _) []) = ((fromIntegral w / 2, 0.5), [])
layoutEdges (Layout gs _ [l@(Layout _ _ _)]) = (t, (t, (tx, ty + 1)) : map ↙
    ↳ translate es)
  where
    t = (fromIntegral (length gs) / 2, 0.5)
    ((tx, ty), es) = layoutEdges l
    translate ((x0, y0), (x1, y1)) = ((x0, y0 + 1), (x1, y1 + 1))
layoutEdges (Layout gs _ [l@(Layout _ (lw, _) _), r]) = (t, (t, (lx, ly + 1)) : ↙
  ↳ (t, (rx + fromIntegral lw + 1, ry + 1)) : map translateL ls ++ map ↙
  ↳ translateR rs)
  where
    t = (fromIntegral lw + fromIntegral (length gs) / 2, 0.5)
    ((lx, ly), ls) = layoutEdges l
    ((rx, ry), rs) = layoutEdges r
    translateL ((x0, y0), (x1, y1)) = ((x0, y0 + 1), (x1, y1 + 1))
    translateR ((x0, y0), (x1, y1)) = ((x0 + fromIntegral lw + 1, y0 + 1), (x1 +↙
      ↳  fromIntegral lw + 1, y1 + 1))
layoutEdges (Layout gs _ [l@(Layout _ (lw, _) _), m@(Layout _ (mw, _) _), r]) =
  (t, (t, (lx, ly + 1)) :
      (t, (mx + fromIntegral lw + 1, my + 1)) :
      (t, (rx + fromIntegral lw + 1 + fromIntegral mw + 1, ry + 1)) :
      map translateL ls ++ map translateM ms ++ map translateR rs)
  where
    t = (fromIntegral lw + fromIntegral (length gs) / 2, 0.5)
    ((lx, ly), ls) = layoutEdges l
    ((mx, my), ms) = layoutEdges m
    ((rx, ry), rs) = layoutEdges r
    translateL ((x0, y0), (x1, y1)) = ((x0, y0 + 1), (x1, y1 + 1))
    translateM ((x0, y0), (x1, y1)) = ((x0 + fromIntegral lw + 1, y0 + 1), (x1 +↙
      ↳  fromIntegral lw + 1, y1 + 1))
    translateR ((x0, y0), (x1, y1)) = ((x0 + fromIntegral lw + 1 + fromIntegral ↙
      ↳ mw + 1, y0 + 1), (x1 + fromIntegral lw + 1 + fromIntegral mw + 1, y1 +↙
      ↳  1))

glyphsU :: U -> [Glyph]
glyphsU Neg  = "-"
glyphsU LNot = "!"
glyphsU BNot = "~"

glyphsB :: B -> [Glyph]
glyphsB Add = "+"
glyphsB Sub = "-"
glyphsB Mul = "*"
glyphsB Div = "/"
glyphsB Mod = "%"
glyphsB BAnd = "&"
glyphsB LAnd = "&&"
glyphsB BOr = "|"
glyphsB LOr = "||"
glyphsB XOr = "^"
glyphsB ShL = "<<"
glyphsB ShR = ">>"
glyphsB Lt = "<"
glyphsB Gt = ">"

pretty :: Layout -> [[(Glyph, Float)]]
pretty l@(Layout _ (w, h) _) = map (([space] ++).(++ [space])) $ [replicate w ↙
  ↳ space] ++ fst (pretty' h l [0..]) ++ [replicate w space] where space = (' ↙
  ↳ ', -1)
pretty' :: Int -> Layout -> [Float] -> ([[(Glyph, Float)]], [Float])
pretty' _ _ [] = error "pretty'"
pretty' h (Layout s (w, _) ls) (c:cs) = (take h (take w (replicate x space ++ (↙
    ↳ zip s (repeat c)) ++ repeat space) : combine [space] gs ++ repeat (↙
    ↳ replicate w space)), cs')
  where
    (gs, cs') = maps (pretty' (h - 1)) cs ls
    space = (' ', c)
    x = case ls of
          (Layout _ (w', _) _):_:_ -> w'
          _ -> 0

maps :: (Layout -> [Float] -> ([[(Glyph, Float)]], [Float])) -> [Float] -> [↙
    ↳ Layout] -> ([[[(Glyph, Float)]]], [Float])
maps _ cs [] = ([], cs)
maps p cs (l:ls) =
  let (g, cs') = p l cs
      (gs, cs'') = maps p cs' ls
  in  (g:gs, cs'')

combine :: [(Glyph, Float)] -> [[[(Glyph, Float)]]] -> [[(Glyph, Float)]]
combine space = map (intercalate space) . transpose

glyphMap :: [(Glyph, [Float])]
glyphMap = [ (g, [x/8, y/4]) | (gs, y) <- ["01234567", "89-~!+*/", "%^&|<>?:", "↙
    ↳ t "] `zip` [0..], (g, x) <- gs `zip` [0..] ]

uploadGlyphs :: [[(Glyph, Float)]] -> IO (Int, Int)
uploadGlyphs gss@(gs:_) = do
  let w = length gs
      h = length gss
      xyzs = concat . mapMaybe (\(g, z) -> fmap (++[z]) $ g `lookup` glyphMap) .↙
          ↳ concat $ gss
  withArray xyzs $ glTexImage2D GL_TEXTURE_RECTANGLE 0 (fromIntegral GL_RGB32F) ↙
      ↳ (fromIntegral w) (fromIntegral h) 0 GL_RGB GL_FLOAT
  return (w, h)
uploadGlyphs _ = return (0, 0)

toTexture :: Layout -> IO (Int, Int)
toTexture = uploadGlyphs . pretty

setupGL :: IO (GLuint, GLuint)
setupGL = do
  [t0, t1] <- withArray [0,0] $ \p -> glGenTextures 2 p >> peekArray 2 p
  glActiveTexture GL_TEXTURE1
  glBindTexture GL_TEXTURE_2D t1
  let width = 1024
      height = 512
      bytes = width * height * 3
  withBinaryFile "glyphs.raw" ReadMode $ \h -> allocaBytes bytes $ \p -> do
    _ <- hGetBuf h p bytes
    glTexImage2D GL_TEXTURE_2D 0 (fromIntegral GL_RGB) (fromIntegral width) (↙
        ↳ fromIntegral height) 0 GL_RGB GL_UNSIGNED_BYTE p
  glGenerateMipmap GL_TEXTURE_2D
  glTexParameteri GL_TEXTURE_2D GL_TEXTURE_MIN_FILTER (fromIntegral ↙
      ↳ GL_LINEAR_MIPMAP_LINEAR)
  glTexParameteri GL_TEXTURE_2D GL_TEXTURE_MAG_FILTER (fromIntegral GL_LINEAR)
  glTexParameteri GL_TEXTURE_2D GL_TEXTURE_WRAP_S (fromIntegral GL_CLAMP_TO_EDGE↙
      ↳ )
  glTexParameteri GL_TEXTURE_2D GL_TEXTURE_WRAP_T (fromIntegral GL_CLAMP_TO_EDGE↙
      ↳ )
  glActiveTexture GL_TEXTURE0
  glBindTexture GL_TEXTURE_RECTANGLE t0
  glTexParameteri GL_TEXTURE_RECTANGLE GL_TEXTURE_MIN_FILTER (fromIntegral ↙
      ↳ GL_NEAREST)
  glTexParameteri GL_TEXTURE_RECTANGLE GL_TEXTURE_MAG_FILTER (fromIntegral ↙
      ↳ GL_NEAREST)
  glTexParameteri GL_TEXTURE_RECTANGLE GL_TEXTURE_WRAP_S (fromIntegral ↙
      ↳ GL_CLAMP_TO_EDGE)
  glTexParameteri GL_TEXTURE_RECTANGLE GL_TEXTURE_WRAP_T (fromIntegral ↙
      ↳ GL_CLAMP_TO_EDGE)
  prog <- glCreateProgram
  frag <- glCreateShader GL_FRAGMENT_SHADER
  src <- readFile "expr.frag"
  withCString src $ \p -> with p $ \pp -> glShaderSource frag 1 pp nullPtr
  glCompileShader frag
  hPutStrLn stderr =<< shaderLog frag
  glAttachShader prog frag
  glLinkProgram prog
  hPutStrLn stderr =<< programLog prog
  glUseProgram prog
  withCString "glyphs" $ \s -> glGetUniformLocation prog s >>= \l -> glUniform1i↙
      ↳ l 1
  withCString "expression" $ \s -> glGetUniformLocation prog s >>= \l -> ↙
      ↳ glUniform1i l 0
  prog2 <- glCreateProgram
  frag2 <- glCreateShader GL_FRAGMENT_SHADER
  src2 <- readFile "stroke.frag"
  withCString src2 $ \p -> with p $ \pp -> glShaderSource frag2 1 pp nullPtr
  glCompileShader frag2
  hPutStrLn stderr =<< shaderLog frag2
  glAttachShader prog2 frag2
  glLinkProgram prog2
  hPutStrLn stderr =<< programLog prog2
  return (prog, prog2)

draw :: (GLuint, GLuint) -> E -> IO ()
draw (prog, prog2) e = do
  let l = layout e
      (_, es) = layoutEdges l
  (w, h') <- toTexture l
  let h  = 1.5 * fromIntegral h'
      w0 = 32 * h / 9
      h0 = 9 * fromIntegral w / 32
      x0 = negate (w0 - fromIntegral w) / 2
      x1 = x0 + w0
      y0 = negate (h0 - h) / 2
      y1 = y0 + h0
  if 9 * fromIntegral w > 32 * h
    then do
      glDisable GL_BLEND
      glUseProgram prog
      quad (0, fromIntegral w) (y1, y0)
      glEnable GL_BLEND
      glBlendFunc GL_DST_COLOR GL_ZERO -- multiplicative
      glUseProgram prog2
      glBegin GL_QUADS
      mapM_ (edge (0, fromIntegral w) (y1, y0)) es
      glEnd
    else do
      glDisable GL_BLEND
      glUseProgram prog
      quad (x0, x1) (h*1.5, 0)
      glEnable GL_BLEND
      glBlendFunc GL_DST_COLOR GL_ZERO -- multiplicative
      glUseProgram prog2
      glBegin GL_QUADS
      mapM_ (edge (x0, x1) (h*1.5, 0)) es
      glEnd
  where
    v vx vy tx ty = glTexCoord2f tx ty >> glVertex2f vx vy
    quad (x0, x1) (y0, y1) = do
      glBegin GL_QUADS
      v (-1) (-1) x0 y0
      v (-1)   1  x0 y1
      v   1    1  x1 y1
      v   1  (-1) x1 y0
      glEnd
    edge (x0, x1) (y0, y1) ((u0, v0), (u1, v1)) = do
      let p0 = ( u0+1      - x0) / (x1 - x0) * 2 - 1
          q0 = ((v0+1)*1.5 - y0) / (y1 - y0) * 2 - 1
          p1 = ( u1+1      - x0) / (x1 - x0) * 2 - 1
          q1 = ((v1+1)*1.5 - y0) / (y1 - y0) * 2 - 1
          dp' = p1 - p0
          dq' = q1 - q0
          d = sqrt (dp' * dp' + dq' * dq') * (x1 - x0)
          dp = 2 * 0.09 * ( dq') / d
          dq = 2 * 0.32 * (-dp') / d
      v (p0 - dp) (q0 - dq) 0 0
      v (p0 + dp) (q0 + dq) 0 1
      v (p1 + dp) (q1 + dq) 1 1
      v (p1 - dp) (q1 - dq) 1 0

programLog :: GLuint -> IO String
programLog prog = do
  l <- with 0 $ \p -> glGetProgramiv prog GL_INFO_LOG_LENGTH p >> peek p
  allocaBytes (fromIntegral l) $ \p -> with 0 $ \q -> do
    glGetProgramInfoLog prog (fromIntegral l) q p
    m <- peek q
    peekCStringLen (p, fromIntegral m)

shaderLog :: GLuint -> IO String
shaderLog prog = do
  l <- with 0 $ \p -> glGetShaderiv prog GL_INFO_LOG_LENGTH p >> peek p
  allocaBytes (fromIntegral l) $ \p -> with 0 $ \q -> do
    glGetShaderInfoLog prog (fromIntegral l) q p
    m <- peek q
    peekCStringLen (p, fromIntegral m)

pngFilename :: Int -> String
pngFilename n = "./v/" ++ (reverse . take 8 . (++ repeat '0') . reverse . show) ↙
    ↳ n ++ ".png"

captureToPNG :: String -> IO ()
```

```haskell
captureToPNG f = do
  let stride = formatStrideForWidth FormatRGB24 videoW
  allocaBytes (videoH * stride) $ \p -> do
    glPixelStorei GL_PACK_ROW_LENGTH (fromIntegral $ stride `div` 4)
    glReadPixels 0 0 (fromIntegral videoW) (fromIntegral videoH) GL_BGRA GL_UNSIGNED_BYTE p
    glPixelStorei GL_PACK_ROW_LENGTH 0
    let q = p `plusPtr` ((videoH - 1) * stride)
    withImageSurfaceForData q FormatRGB24 videoW videoH (-stride) $ \s -> do
      surfaceWriteToPNG s f
```