prof2pretty

Claude Heiland-Allen

2010–2018

Contents

1 data/prof2pretty.css

```
#ui { width: 200px; position: fixed; }
#ui .left { float: left; text-align: center; }
#ui .left p { width: 90px; }
#ui .left div { margin: auto; }
#ui-limit, #ui-gamma { height: 500px; }
#main { margin-left: 220px; }
#main a[href] { color: black; font-weight: bold; text-decoration: none; }
#main .line-number { font-size: 80%; vertical-align: middle; }
```

2 data/prof2pretty.js

```
var mode = 0, low = 0, high = 1, gamma = 1;

function colour(x) {
  return 'hsl(' + Math.round(240 * x) + ',100%,' + Math.round(50 + 25 * x) + '%)';
}

function colourize() {
  var prefix = [ 'top-', '' ];
  for (p in prefix) {
    for (id in profile) {
      var e = document.getElementById(prefix[p] + id);
      if (e) {
        var s = e.style;
        var v = profile[id][mode];
        if (low < v && v <= high) {
          s.display = 'inline-block';
          s.padding = '2px';
          s.backgroundColor = colour(Math.pow(1 - (profile[id][mode] - low) / (high - low), gamma));
```

```
        } else {
          s.display = '';
          s.padding = '';
          s.backgroundColor = '';
        }
      }
    }
  }
}

function go() {
  var options = new Object();
  var q = '' + this.location;
  q = q.substring(q.indexOf('?') + 1);
  if (q.indexOf('#') > -1) {
    q = q.substring(0, q.indexOf('#'));
  }
  if (q.length >= 1) {
    var kv = new Object();
    var n = 1;
    while (q.indexOf('&') > -1) {
      kv[n] = q.substring(0, q.indexOf('&'));
      q = q.substring(q.indexOf('&') + 1 );
      n = n + 1;
    }
    kv[n] = q;
    for (var i = 0; i <= n; i = i + 1) {
      var k = (''+kv[i]).substring(0, (''+kv[i]).indexOf('='));
      var v = (''+kv[i]).substring((''+kv[i]).indexOf('=') + 1);
      options[k] = v;
    }
  }
  var def = function(k,v) {
    if (null == options[k]) { return v; } else { return options[k]; }
  };
  var modes     = [ 'ticks', 'rticks', 'bytes', 'rbytes' ];
  var modeNames = { 'ticks': 0, 'rticks': 1, 'bytes': 2 , 'rbytes': 3 };
  mode  = modeNames[def('mode', 'ticks')]; if (mode == null) { mode = 0; };
  low   = Math.min(Math.max(parseFloat(def('low',   0)), 0),  1);
  high  = Math.min(Math.max(parseFloat(def('high',  1)), 0),  1);
  gamma = Math.min(Math.max(parseFloat(def('gamma', 1)), 0.1), 10);
  document.getElementById('ui-' + modes[mode]).setAttributeNS(null, 'checked', '↙
    ↳ checked');
  $('#ui-mode').buttonset();
  $('#ui-ticks').click(function() { mode = modeNames['ticks']; colourize(); });
  $('#ui-bytes').click(function() { mode = modeNames['bytes']; colourize(); });
  $('#ui-limit').slider({ orientation: 'vertical', min: 0, max: 1000, step: 1, ↙
    ↳ range: true,
    values: [1000 * (1 - high), 1000 * (1 - low)],
    change: function(event, ui) {
      low = 1 - ui.values[1] / 1000.0; high = 1 - ui.values[0] / 1000.0;
      colourize();
    }
  });
  $('#ui-gamma').slider({ orientation: 'vertical', min: 0, max: 1000, step: 1,
    value: 1000 * (Math.log(gamma)/5 + 0.5),
    change: function(event, ui) {
```

```
      gamma = Math.exp(5 * (ui.value / 1000.0 - 0.5));
      colourize();
    }
  });
  colourize();
}
```

3 example/test.hs

```
-- The Computer Language Benchmarks Game
-- http://shootout.alioth.debian.org/
-- contributed by Bryan O'Sullivan
-- modified by Eugene Kirpichov: pidgits only generates
-- the result string instead of printing it. For some
-- reason, this gives a speedup.

import System.Environment (getArgs)

pidgits n = 0 % (0 # (1,0,1)) where
 i%ds
  | i >= n = []
  | True = (concat h ++ "\t:" ++ show j ++ "\n") ++ j%t
  where k = i+10; j = min n k
        (h,t) | k > n = (take (n`mod`10) ds ++ replicate (k-n) " ",[])
              | True = splitAt 10 ds
 j # s | n>a || r+n>=d = k # t
       | True = show q : k # (n*10,(a-(q*d))*10,d)
  where k = j+1; t@(n,a,d)=k&s; (q,r)=(n*3+a)`divMod`d
 j&(n,a,d) = (n*j,(a+n*2)*y,d*y) where y=(j*2+1)

main = putStr.pidgits.read.head =<< getArgs
```

4 .gitignore

```
dist
-
```

5 LICENSE

```
Copyright (c) 2012, Claude Heiland-Allen

All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

    * Redistributions of source code must retain the above copyright
      notice, this list of conditions and the following disclaimer.

    * Redistributions in binary form must reproduce the above
      copyright notice, this list of conditions and the following
      disclaimer in the documentation and/or other materials provided
      with the distribution.

    * Neither the name of Claude Heiland-Allen nor the names of other
      contributors may be used to endorse or promote products derived
```

```
      from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

6 prof2pretty.cabal

```
name:                prof2pretty
version:             0.1.0.0
synopsis:            generate pretty source from time/allocation profiles
description:
  prof2pretty is an enhanced rewrite of visual-prof that works with
  recent GHC RTS.  sccpragmabomb adds SCC pragmas encoding source
  location.  prof2pretty extracts those SCCs from the profiling output
  to annotate the sources using HTML + CSS + JavaScript.
  .
  Usage:
  .
  > cd example
  > ghc -prof -F -pgmF=sccpragmabomb test.hs
  > cd ..
  > ./example/test +RTS -P -RTS 31416
  > prof2pretty --standalone --source=example/ test.prof
  > sensible-browser test.prof.html
  .
  Example output: <https://mathr.co.uk/prof2pretty/prof2pretty-0.1.0.0.html>

homepage:            https://code.mathr.co.uk/prof2pretty
license:             BSD3
license-file:        LICENSE
author:              Claude Heiland-Allen
maintainer:          claude@mathr.co.uk
category:            Development, Profiling
build-type:          Simple
cabal-version:       >=1.8

extra-source-files:
  example/test.hs

data-dir: data
data-files:
  prof2pretty.js
  prof2pretty.css
  jquery-1.7.2.min.js
  jquery-ui-1.8.21.custom.min.js
  jquery-ui-1.8.21.custom.css

executable prof2pretty
```

```
    hs-source-dirs: src
    main-is: prof2pretty.hs
    other-modules: Paths_prof2pretty
    build-depends:
      base >= 4 && < 5,
      containers,
      filepath,
      semigroups,
      zenc
    ghc-options: -Wall

executable sccpragmabomb
    hs-source-dirs: src
    main-is: sccpragmabomb.hs
    build-depends:
      base >= 4 && < 5,
      haskell-src-exts,
      uniplate,
      zenc
    ghc-options: -Wall

source-repository head
    type:                git
    location:            https://code.mathr.co.uk/prof2pretty.git

source-repository this
    type:                git
    location:            https://code.mathr.co.uk/prof2pretty.git
    tag:                 v0.1.0.0
```

7 Setup.hs

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

8 src/prof2pretty.hs

```
module Main (main) where

import Control.Monad (forM_, unless)
import Data.Char (isSpace)
import Data.List (intercalate, isPrefixOf)
import Data.Maybe (listToMaybe, mapMaybe)
import Data.Map (Map)
import qualified Data.Map as M
import Data.Semigroup (Max(Max), (<>), Sum(Sum), mappend, mconcat)
import Data.Set (Set)
import qualified Data.Set as S
import Data.Tree (Tree(..), Forest, flatten)
import System.Environment (getArgs)
import System.Exit (exitFailure)
import System.FilePath ((</>), (<.>), dropExtension)
import System.IO (hPutStr, hPutStrLn, stderr, withFile, IOMode(WriteMode))
import Text.Encoding.Z (zDecodeString, zEncodeString)

import Paths_prof2pretty (getDataFileName)
```

```
20
type SrcLoc  = (Int, Int)
type SrcSpan = (SrcLoc, SrcLoc)
type FileSpan = (String, SrcSpan)

25 decodeSCC :: String -> Maybe FileSpan
decodeSCC s = case (reverse . take 5 . reverse . split '-') s of
  z:is -> case mapMaybe readMay is of
    [l0,c0,l1,c1] -> Just (zDecodeString z, ((l0, c0), (l1, c1)))
    _ -> Nothing
30  _ -> Nothing

notNull :: [a] -> Bool
notNull = not . null

35 deepMapForest :: (Forest a -> Forest a) -> Tree a -> Tree a
deepMapForest f (Node label forest) = Node label (f (map (deepMapForest f) ↙
    ↳ forest))

split :: Eq a => a -> [a] -> [[a]]
split _ [] = []
40 split x xs = let (ys, zs) = break (x ==) xs in ys : split x (drop 1 zs)

readMay :: Read a => String -> Maybe a
readMay s = case reads s of
  [(x, "")] -> Just x
45  _ -> Nothing

preprocess :: String -> ([String], [String])
preprocess
  = (concatMap (take 1 . words) *** (dropWhile null . dropWhile notNull . ↙
      ↳ dropWhile null))
50  . break null . drop 9 . lines

data Entry = Entry
  { eSCC     :: String
  , eTicks   :: !Integer
55  , eBytes   :: !Integer
  }
  deriving (Show)

parseLine :: String -> (Int, Entry)
60 parseLine s =
  let (indent, rest) = span isSpace s
      [scc,_,_,_,_,_,_,_,ticks,bytes] = words rest
  in  (length indent, Entry scc (read ticks) (read bytes))

65 type Entries = [(Int, Entry)]

collate :: Entries -> Tree Entry
collate input =
  let ([[result]], []) = collate' (-1) [] input
70  in  deepMapForest reverse result

collate' :: Int -> [Forest Entry] -> Entries -> ([Forest Entry], Entries)
collate' pdepth stack [] = (adjust pdepth stack , [])
collate' pdepth stack ((depth, entry) : rest) =
```

```
    let siblings : parents = adjust (pdepth - depth) stack
        node = Node{ rootLabel = entry, subForest = [] }
    in  collate' depth ((node : siblings) : parents) rest

adjust :: Int -> [Forest Entry] -> [Forest Entry]
adjust n t = case compare n 0 of
  LT -> adjust (n + 1) ([] : t)
  EQ ->                  t
  GT -> adjust (n - 1) (case t of
    (ts:(t':ts'):ts'') -> (t'{ subForest = subForest t' ++ ts } : ts') : ts''
    _ -> error $ "prof2pretty.adjust: " ++ show (n, t))

parseEntries :: [String] -> Tree Entry
parseEntries = collate . map parseLine

type Proportion = ((Double, Double), (Double, Double))

proportion :: Tree Entry -> Tree (Entry, Proportion)
proportion t =
  let u@Node{ rootLabel = (_, (Sum totalTicks, Sum totalBytes)) } = up t
  in  down totalTicks totalBytes totalTicks totalBytes u
  where
    me e = (Sum (eTicks e), Sum (eBytes e))
    up Node{ rootLabel = e, subForest = [] } =
          Node{ rootLabel = (e, me e), subForest = [] }
    up Node{ rootLabel = e, subForest = ts } =
      let us = map up ts
          them = mconcat (map (snd . rootLabel) us)
      in  Node{ rootLabel = (e, me e `mappend` them), subForest = us }
    down totalTicks totalBytes parentTicks parentBytes
      Node{ rootLabel = (e, (Sum ticks, Sum bytes)), subForest = fs } =
        let ticky = (eTicks e /// totalTicks, ticks /// parentTicks)
            bytey = (eBytes e /// totalBytes, bytes /// parentBytes)
            local = (ticky, bytey)
        in  Node
              { rootLabel = (e, local)
              , subForest = map (down totalTicks totalBytes ticks bytes) fs
              }

(///) :: Integer -> Integer -> Double
a /// b = fromIntegral a / if b > 0 then fromIntegral b else 1

(***) :: (a -> c) -> (b -> d) -> (a, b) -> (c, d)
f *** g = \(a, b) -> (f a, g b)

second :: (b -> d) -> (a, b) -> (a, d)
second g = id *** g

compose :: [a -> a] -> a -> a
compose = foldr (.) id

type Profile = ([String], Map String Proportion)

collect :: Tree (Entry, Proportion) -> Map String Proportion
collect = fmap unM2 . M.fromListWith (<>) . map (eSCC *** m2) . flatten
  where
    unM (Sum a, Max b) = (a, b)
```

```
    unM2 (a, b) = (unM a, unM b)
    m2 (a, b) = (m a, m b)
    m (a, b) = (Sum a, Max b)

parseProfile :: String -> Profile
parseProfile = second (collect . proportion . parseEntries) . preprocess

characterize :: String -> [[String]]
characterize = map (map (:[])) . lines

decharacterize :: [[String]] -> String
decharacterize = unlines . map concat

xmlSafe :: String -> String
xmlSafe = concatMap x
  where
    x '&' = "&amp;"
    x '<' = "&lt;"
    x '>' = "&gt;"
    x  c  = [c]

wrapLine :: Int -> [a] -> [a] -> [[[a]]] -> [[[a]]]
wrapLine line pre post
  = prependAt (line,         1) pre
  . appendAt  (line, maxBound) post

prependAt :: (Int, Int) -> [a] -> [[[a]]] -> [[[a]]]
prependAt loc str = at2 loc (str ++)

appendAt :: (Int, Int) -> [a] -> [[[a]]] -> [[[a]]]
appendAt (l, c) str
  | c == 0    = appendAt (l - 1, maxBound) str
  | otherwise = at2      (l, c - 1)    (++ str)

at2 :: (Int, Int) -> ([a] -> [a]) -> [[[a]]] -> [[[a]]]
at2 (line, col) f = at1 line (at1 col f)

at1 :: Int -> ([a] -> [a]) -> [[a]] -> [[a]]
at1 n f xs = case splitAt (n - 1) xs of
  (pre, [     ]) -> pre ++ [f []]
  (pre, at:post) -> pre ++ [f at] ++ post

annotate :: [String] -> String -> String -> String
annotate prof file =
    decharacterize . (foldr f `flip` prof) . map (map xmlSafe) . characterize
  where
    f scc = case decodeSCC scc of
      Nothing -> id
      Just (file', (start@(sl, _), end'@(el', ec')))
        | file /= file' -> id
        | sl == el  -> prependAt start open . appendAt end close
        | otherwise -> prependAt start open . appendAt (sl, maxBound) close
                     . compose [ wrapLine l open close | l <- [sl + 1 .. el - 1]↙
                       ↳ ]
                     . prependAt (el, 1) open . appendAt end close
        where
          end@(el, _)
```

```
            | ec' == 1  = (el' - 1, maxBound)
            | otherwise = end'
          open  = "<span id=\"" ++ scc ++ "\">"
          close = "</span>"

readProfile :: String -> IO Profile
readProfile f = parseProfile `fmap` readFile f

profileFiles :: Profile -> Set String
profileFiles = S.fromList . mapMaybe (fmap fst . decodeSCC) . M.keys . snd

profileSCCs :: Profile -> [String]
profileSCCs = M.keys . snd

addLineNumbers :: String -> String -> String
addLineNumbers file = unlines . map (\(n, s) ->
  "<span class=\"line-number\">" ++ replicate (6 - length (show n)) ' '
  ++ "<a id=\"" ++ zEncodeString file ++ "-" ++ show n ++ "\" "
  ++ "href=\"#" ++ zEncodeString file ++ "-" ++ show n ++ "\">"
  ++ show n ++ "</a> </span>" ++ s) . zip [(1 :: Integer)..] . lines

main :: IO ()
main = do
  args <- getArgs
  let standalone = "--standalone" `elem` args
      Just source = listToMaybe . (++ [""]) . map (drop (length sourcePrefix)) ↙
          ↳ . filter (sourcePrefix `isPrefixOf`) $ args
        where sourcePrefix = "--source="
      mprofF = listToMaybe . filter (not . ("--" `isPrefixOf`)) $ args
  profF <- case mprofF of
    Just p -> return p
    _ -> hPutStrLn stderr "usage: prof2pretty [--standalone] [--source='/path/to↙
        ↳ /src'] executable.prof" >> exitFailure
  prof <- readProfile profF
  withFile (profF <.> "html") WriteMode $ \h -> do
    let files = S.toList $ profileFiles prof
        title = xmlSafe (dropExtension profF)
        profFJSFile = profF ++ ".js"
        profFJS =
          "var profile =\n{" ++ intercalate "\n,"
            [ show s ++ ":" ++ show [a, b, c, d]
            | (s, ((a, b), (c, d))) <- M.toList (snd prof)
            , a + b + c + d > 0
            ] ++ "\n};\n"
    unless standalone $ writeFile profFJSFile profFJS
    profCSSFile <- getDataFileName "prof2pretty.css"
    profJSFile <- getDataFileName "prof2pretty.js"
    jqueryUICSSFile <- getDataFileName "jquery-ui-1.8.21.custom.css"
    jqueryJSFile <- getDataFileName "jquery-1.7.2.min.js"
    jqueryUIJSFile <- getDataFileName "jquery-ui-1.8.21.custom.min.js"
    let readFile' = if standalone then readFile else const (return "")
    profCSS <- readFile' profCSSFile
    profJS <- readFile' profJSFile
    jqueryUICSS <- readFile' jqueryUICSSFile
    jqueryJS <- readFile' jqueryJSFile
    jqueryUIJS <- readFile' jqueryUIJSFile
    let css file text
```

```
          | standalone = "<style type=\"text/css\">/* <![CDATA[ */\n" ++ text ++ "\n/* ]]> */</style>"
          | otherwise  = "<link type=\"text/css\" href=\"" ++ xmlSafe file ++ "\" rel=\"stylesheet\" />"
        js file text
          | standalone = "<script type=\"text/javascript\">/* <![CDATA[ */\n" ++ text ++ "\n/* ]]> */</script>"
          | otherwise  = "<script type=\"text/javascript\" src=\"" ++ xmlSafe file ++ "\"></script>"
    hPutStr h . unlines $
      [ "<!DOCTYPE html>\n<html><head><title>" ++ title ++ "</title>"
      , css jqueryUICSSFile jqueryUICSS
      , js jqueryJSFile jqueryJS
      , js jqueryUIJSFile jqueryUIJS
      , css profCSSFile profCSS
      , js profFJSFile profFJS
      , js profJSFile profJS
      , "</head><body onload=\"go()\">"
      , "<div id=\"ui\">\n<div class=\"left\"><p>mode</p><form id=\"ui-mode\">"
      , "<input type=radio id=\"ui-ticks\" name=\"ui-mode\" /><label for=\"ui-ticks\">ticks</label>"
      , "<input type=radio id=\"ui-bytes\" name=\"ui-mode\" /><label for=\"ui-bytes\">bytes</label>"
      , "</form></div>"
      , "<div class=\"left\"><p>limit</p><div id=\"ui-limit\"></div></div>"
      , "<div class=\"left\"><p>gamma</p><div id=\"ui-gamma\"></div></div>"
      , "</div>\n<div id=\"main\">\n<h1>" ++ title ++ "</h1>\n<h2>Summary</h2><ul>"
      ] ++
      [ case decodeSCC scc of
          Nothing ->
            "<li><a id=\"top-" ++ xmlSafe scc ++ "\">" ++ xmlSafe scc ++ "</a></li>"
          Just (file, ((sl, sc), (el, ec))) ->
            "<li><a id=\"top-" ++ xmlSafe scc ++ "\" href=\"#" ++ xmlSafe scc ++ "\">" ++
            xmlSafe file ++ ":" ++ show sl ++ ":" ++ show sc ++ "-" ++ show el ++ ":" ++ show ec ++
            "</a></li>"
      | scc <- fst prof
      ] ++ [ "</ul>", "<h2>Details</h2>" ]
    forM_ files $ \file -> do
      hPutStr h $ "<div id=\"" ++ zEncodeString file ++ "\"><h3>" ++ xmlSafe file ++ "</h3>\n<pre>"
      hPutStr h . addLineNumbers file . annotate (profileSCCs prof) file =<< readFile (source </> file)
      hPutStr h $ "</pre></div>\n"
    hPutStrLn h "</div>\n</body></html>"
```

9 src/sccpragmabomb.hs

```
module Main (main) where

import Language.Haskell.Exts.Annotated
  ( Module
  , Exp(SCCPragma, Paren)
  , SrcSpanInfo
```

```
    , srcInfoSpan
    , SrcSpan(..)
    , fromParseResult
    , parseFileContentsWithMode
    , defaultParseMode
    , parseFilename
    , ann
    , prettyPrintWithMode
    , defaultMode
    , linePragmas
    )
import Data.Generics.Uniplate.Data (transformBi)
import Control.Exception (handle, SomeException(..))
import System.Environment (getArgs)
import System.Exit (exitFailure)
import System.IO (hPutStrLn, stderr)
import Text.Encoding.Z (zEncodeString)

sccPragmaBomb :: String -> Module SrcSpanInfo -> Module SrcSpanInfo
sccPragmaBomb s = transformBi (addSCC s)

addSCC :: String -> Exp SrcSpanInfo -> Exp SrcSpanInfo
addSCC s e =
  let f = stripSCC e
      a = ann f
  in  Paren a (SCCPragma a (sccFromSrcSpan s (srcInfoSpan a)) f)

stripSCC :: Exp SrcSpanInfo -> Exp SrcSpanInfo
stripSCC (SCCPragma _ _ e) = e
stripSCC e = e

sccFromSrcSpan :: String -> SrcSpan -> String
sccFromSrcSpan s (SrcSpan
    { srcSpanStartLine = l0
    , srcSpanStartColumn = c0
    , srcSpanEndLine = l1
    , srcSpanEndColumn = c1
    }) = zEncodeString s ++ concatMap (('-':) . show) [l0,c0,l1,c1]

parse :: String -> String -> Module SrcSpanInfo
parse name = fromParseResult . parseFileContentsWithMode defaultParseMode{ ↙
    ↳ parseFilename = name }

main :: IO ()
main = handle (\(SomeException e) -> hPutStrLn stderr ("sccpragmabomb: " ++ show↙
    ↳  e) >> exitFailure) $ do
  [originalFile, sourceFile, destinationFile] <- getArgs
  writeFile destinationFile . prettyPrintWithMode defaultMode{ linePragmas = ↙
      ↳ True } . sccPragmaBomb originalFile . parse originalFile =<< readFile ↙
      ↳ sourceFile
```