The Next 700 Probabilistic Programming Languages

Jeffrey Mark Siskind

PURDUE UNIVERSITY

Thursday 22 January 2015

Brooks Paige Frank Wood

This research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-10-2-0060. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either express or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes, notwithstanding any copyright notation herein.

Probabilistic C

Probabilistic C Boilerplate

```
/* ERPs */
long poisson_rng(double rate);
double normal_lnp(double x, double mean, double variance);
/* plus more _rng and _lnp */

/* directives */
void observe(const double ln_p);
void predict_value(const char *name, const double value);
```

An Example in Probabilistic C

```
#include <probabilistic.h>

int main(int argc, char *argv[]) {
  long a = poisson_rng(100.0)-100;
  long b = poisson_rng(100.0)-100;
  observe(normal_lnp(7.0, (double)(a+b), 0.00001));
  predict_value("a", (double)a);
  predict_value("b", (double)b);
}
```

Probabilistic SCHEME

Probabilistic STALIN Boilerplate

```
;;; ERPs
(define poisson-rng
 (foreign-procedure (double) long "poisson_rng" "probabilistic"))

(define normal-lnp
 (foreign-procedure (double double double) double "normal_lnp" "probabilistic"))
;;; plus more -rng and -lnp

;;; directives
(define observe (foreign-procedure (double) void "observe" "probabilistic"))

(define predict-value
 (foreign-procedure (char* double) void "predict_value" "probabilistic"))

;;; necessary boilerplate
(vector-ref argv 0)
```

An Example in Probabilistic STALIN

```
(include "probabilistic")

(define a (- (poisson-rng 100.0) 100))
(define b (- (poisson-rng 100.0) 100))
(observe (normal-lnp 7.0 (exact->inexact (+ a b)) .00001))
(predict-value "a" (exact->inexact a))
(predict-value "b" (exact->inexact b))
```

Probabilistic SML

Probabilistic MLton C Boilerplate

```c
#include <probabilistic.h>
#include <setjmp.h>

void predict_value_wrapper(unsigned char *name, double value);

extern jmp_buf env;

void return_from_main(int val);
```

Probabilistic MLton C Boilerplate

```c
#include "wrapper.h"

void predict_value_wrapper(unsigned char *name, double value) {
  predict_value(name, value);
}

void return_from_main(int val) {
  longjmp(env, val);
}
```

Probabilistic MLton C Boilerplate

```c
#include <stdlib.h>

jmp_buf env;

int main (int argc, char *argv[]) {
  if (!setjmp(env)) MLton_main(argc, argv);
  return EXIT_SUCCESS;
}
```

Probabilistic MLton SML Boilerplate

```
(* ERPs *)
val poisson_rng = _import "poisson_rng": real->Int64.int;
val normal_lnp = _import "normal_lnp": real*real*real->real;
(* plus more _rng and _lnp *)

(* directives *)
val observe = _import "observe": real->unit;
val predict_value_wrapper =
    _import "predict_value_wrapper": string*real->unit;
fun predict_value (s, x) = predict_value_wrapper (s^"\000", x)

(* necessary utility *)
fun int64ToReal i = Real.fromLargeInt (Int64.toLarge i)

(* necessary boilerplate *)
val return_from_main = _import "return_from_main": int->unit;
```

An Example in Probabilistic MLton

```
val a = (poisson_rng 100.0)-100
val b = (poisson_rng 100.0)-100
val _ = observe (normal_lnp (7.0, (int64ToReal (a+b)), 0.00001))
val _ = predict_value ("a", (int64ToReal a))
val _ = predict_value ("b", (int64ToReal b))
val _ = return_from_main 0
```

Probabilistic HASKELL

Probabilistic GHC C Boilerplate

```
#include <stdlib.h>
#include <probabilistic.h>
#include "sum-equals_stub.h"
```

Probabilistic GHC C Boilerplate

```c
#include "wrapper.h"

int main(int argc, char *argv[]) {
  hs_init(&argc, &argv);
  model();
  /* hs_exit(); */
  return EXIT_SUCCESS;
}
```

Probabilistic GHC HASKELL Boilerplate

```
{-# LANGUAGE ForeignFunctionInterface #-}

module Model where
import Foreign.C
import Foreign.C.String
import Control.Applicative

  -- ERPs
foreign import ccall "poisson_rng" poisson_rng :: CDouble -> IO CInt
foreign import ccall "normal_lnp" normal_lnp ::
  CDouble -> CDouble ->  CDouble -> CDouble
  -- plus more _rng and _lnp

  -- directives
foreign import ccall "observe" observe :: CDouble -> IO ()
foreign import ccall "predict_value" predict_value_wrapper ::
  CString -> CDouble -> IO ()
predict_value string value =
  withCString string (\cstring -> predict_value_wrapper cstring value)

  -- necessary boilerplate
foreign export ccall model :: IO ()
```

An Example in Probabilistic GHC

```
model = do
  a <- (+(-100)) <$> poisson_rng 100.0
  b <- (+(-100)) <$> poisson_rng 100.0
  observe $ normal_lnp 7 (realToFrac (a+b)) 0.00001
  predict_value "a" (realToFrac a)
  predict_value "b" (realToFrac b)
  return ()
```