ProbL

A Probabilistic Modeling Language

Team

Nir Grinberg

Sam Tkach

Diana Liskovich

Andrew Wong

Main Goal

```
input {
        float [] x;
        float [] y;
}
output {
        float a;
        float b;
        float sigma;
}
model {
        y ~ norm(a + b * x, sigma);
}
```

Main Goal

```
input {
        float [] x; float [] y;
}
output {
        float a; float b; float sigma;
}
fun~ norm(float[] x, float[] y): a,b,sigma {
        /* updating equations */
        a = …
}
model {
        a,b,sigma ~ norm(x, y);
}
```

Features

- **Simple** input and output
  - ProbL users can specify what data to use and which parameters to estimate
- The core value of ProbL is **easy and extensible** specification of **statistical models**
- Meant to help statisticians focus on modeling data rather than implementing difficult inference algorithms

Types

- int
- float
- string
- bool
- array
- enum*

Function Definitions

```
fun function_name (type 1 arg1, type2, arg2, ...) : return_type
{
      /* ... function code here ... */
      return output_value;
}

fun~ function_name (type 1 arg1, type2 arg2, ...) : param1, param2, ...
{
      /* ... inference updating equations ... */
}
```

Input/Output

```
input {
        float [] x;
        float [] y;
}
output {
        float a;
        float b;
        float sigma;
}
...
```

Tests

- Testing generated java code

```
import java.util.*;
import java.lang.*;

class program
{
//input

//output

private static Random __rand = new Random(System.currentTimeMillis());

public static void main(String[] args){
System.out.println("hello world");

return ;
}
}
```

- … and output (‘hello world’)

Compiler Pipeline

Lessons Learned

- Writing a compiler is hard!
- Make baby steps!!
- Switching code generator is pretty easy
- Collaboration is key! (Github, Overleaf, Google Docs)