**Goals & Solution**

**Goals**
- Allow BLOG to benefit from progress in Figaro’s inference engine
- Understand how to code open-universe models efficiently in an embedded, functional PPL
- Understand relationship between possible-world and random-evaluation semantics in practice

**Solution**: We developed a compiler BFiT with dynamic memoization techniques to translate a BLOG program to a Figaro program with the same output result.

**Background**

**Possible-World Semantics**: a program with PW semantics defines a probability measure over possible worlds.

**Random-Evaluation Semantics**: a program with RE semantics defines a probability measure over execution traces or partial traces.

**Open Universe Probability Models (OUPMs)**: model uncertainties in the existence and identity of objects and the relations among them.

**Proposed New Syntax**

```scala
class Person extends BaseClass("Person");
class Login extends BaseClass("Login");
CreateObj[Person](Poisson(5));
MakeOrigin[Person,Login]("Owner",(p:Person) => If(Honest(p),Constant(1),Poisson(10)));
```

**Evaluation**

**Theorem**: BFiT always produces a target code in Figaro from a BLOG model with a constant blowup factor in program size.

**Experiment**: Lines of Code in Blog & Figaro

<table>
<thead>
<tr>
<th>Model</th>
<th>Loc. in BLOG</th>
<th>Loc. in Figaro</th>
<th>Model</th>
<th>Loc. in BLOG</th>
<th>Loc. in Figaro</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSI</td>
<td>14</td>
<td>71</td>
<td>Urnball</td>
<td>38</td>
<td>126</td>
</tr>
<tr>
<td>Burglary</td>
<td>22</td>
<td>87</td>
<td>Citation</td>
<td>40</td>
<td>178</td>
</tr>
<tr>
<td>Hurricane</td>
<td>37</td>
<td>168</td>
<td>TugWar</td>
<td>55</td>
<td>278</td>
</tr>
</tbody>
</table>

**Number Statement w/ Origin Function**

```scala
val _Person = Poisson(5);
val _Persons = MakeList(_Person,()=>Select(1.0->new Person));
```

**Dynamic Memorization by BFiT**

```scala
val N_Person = Poisson(5);
val Persons = MakeList(N_Person,()=>Select(1.0->new Person));
```

**Translated Program by BFiT**

```scala
class Login.ORIGIN.Owner:
  val Owner = ORIGIN.Owner;

val n = If(Honest(Owner),Constant(1),Poisson(10));
val L = MakeList(n,Select(1.0->new Login(Owner)));
```

**Goals & Challenge**

**Person-Login Model (BLOG)**

```scala
type Person, Login;

random Boolean Honest(Person x) ~ BooleanDist(0.9);  
origin Person Owner(Login);

random Login A ~ UniformChoice
  ( (x for Login x));

query Honest(Owner(A));
```

**Number Statement**

```scala
Number Statement =>
```

**Number Statement w/ Origin Function**

```scala
Number Statement w/ Origin Function
```

**Number Statement**

```scala
Number Statement
```

**Proposed New Syntax**

```scala
class Person extends BaseClass("Person");
class Login extends BaseClass("Login");
CreateObj[Person](Poisson(5));
MakeOrigin[Person,Login]("Owner",(p:Person) => If(Honest(p),Constant(1),Poisson(10)));
```

**Figure**

- **BFiT: From Possible-World Semantics to Random-Evaluation Semantics in an Open Universe**
- **Goals**: Understand how to code open-universe models efficiently in an embedded, functional PPL.
- **Solution**: Developed a compiler BFiT with dynamic memoization techniques.
- **OUPM Example & Challenge**
  - **Model**: Person-Login Model (BLOG)
  - **Type**: Person, Login
  - **Random Function**: Honest(Person x) ~ BooleanDist(0.9)
  - **Origin Function**: Person Owner(Login)
  - **Number Statement**: Honest(Owner(A))
- **Dynamic Memorization by BFiT**
  - **Create a list to store all the persons**
  - **Use a Map to memoize each random function**
  - **Represent Origin Function as a field**
- **Translated Program by BFiT**
  - **Create a list of logins with the same Owner**
  - **Combine lists to a new List**
- **Evaluation**
  - **Theorem**: BFiT always produces a target code in Figaro from a BLOG model with a constant blowup factor in program size.
  - **Experiment**: Lines of Code in Blog & Figaro
- **Proposed New Syntax**: New syntax for Person and Login classes.