Translating Probabilistic Programs to Factor Graph Grammars

David Chiang, University of Notre Dame
Chung-chieh Shan, Indiana University

A probabilistic program with
1. conditional control flow and
2. unbounded recursion.

A factor graph grammar, which generates a set of factor graphs that together describe the same probability distribution as the program. Exact inference is possible without enumerating the (infinite) set of graphs.

Translating Conditionals and Recursion

1. Conditionals translate to two rules, one for each arm.

```latex
\text{fun } d(x) = \\
\text{case sample } p[x] \text{ of} \\
inl a \Rightarrow \\
unit \\
inr yz \Rightarrow \\
let u = d(fst(yz)) \text{ in} \\
d(snd(yz));
```

2. Functions also translate to rules and can be recursive.

```latex
\text{fun } f(x) = x; \\
f(42)
```

Future Work

- Implement the translation: Currently only outputs LaTeX!
- Implementing FGGs is also future work. In particular, we will implement a sum-product algorithm that outputs a PyTorch computation graph.
- Extend the translation

The translation is not surjective. What kind of program would translate to a rule like the one at left?
A function with multiple outputs, and inputs can depend on outputs (but still acyclic), like a coroutine?