Currently supports highly efficient implementation. This work-in-progress is concerned with universal probabilistic programming languages (PPLs).

- Examples: WebPPL, Anglican, Birch.
- Currently, few PPL systems make use of HPC hardware.
- Examples: LibBi (CUDA, not universal), Pyro (PyTorch), Edward (TensorFlow).
- Our goal: Develop a highly efficient framework for compiling universal probabilistic programs written in a core PPL down to pure CUDA GPGPU code.

Part I: Core Language for Domain-Specific PPLs (Work in progress)

- A small universal PPL based on the lambda calculus.
- A target intermediate language when translating from domain-specific PPLs.
- Also envisioned as an intermediate language for optimization phases.

Part II: PPL Validation and Compilation (Work in progress)

- Rule out higher-order programs that cannot be compiled to the GPU.
- Not allowed: Closures, data structures requiring garbage collection, etc.
- Transform the CorePPL AST into a PCFG.
- Explicitly manage a call stack to support complex control flow involving BBLOCKS and inference algorithms (e.g., resampling in SMC).
- Generate the actual CUDA C++ code, which is later compiled using the CUDA compiler.

Part III: GPGPU Runtime

- Currently supports highly efficient SMC inference using CUDA (also supports OpenMP).
- This implementation has been evaluated on a number of complex models from phylogenetics (with impressive results).
- We are working on adding support for more inference algorithms (e.g., MCMC).

Examples:

```
Part I: Core Language for Domain-Specific PPLs (WIP)
CorePPL AST → PPL DLSL

Part II: PPL Validation and Compilation (WIP)
CorePPL AST → PCFG Block Identification → PCFG

Part III: GPGPU Runtime
PPL DLSL → Parsing and Translation → CorePPL AST → Program Validity Check → PCFG Generation
```

Overview

- This work-in-progress is concerned with universal probabilistic programming languages (PPLs).
- Examples: WebPPL, Anglican, Birch.
- Currently, few PPL systems make use of HPC hardware.
- Examples: LibBi (CUDA, not universal), Pyro (PyTorch), Edward (TensorFlow).
- Our goal: Develop a highly efficient framework for compiling universal probabilistic programs written in a core PPL down to pure CUDA GPGPU code.

Part III: GPGPU Runtime

- Currently supports highly efficient SMC inference using CUDA (also supports OpenMP).
- This implementation has been evaluated on a number of complex models from phylogenetics (with impressive results).
- We are working on adding support for more inference algorithms (e.g., MCMC).