FlatPPL

A Flat Portable Probabilistic Language

Oliver Schulz, Benjamin Cox

oschulz@mpp.mpg.de, bcox@mpp.mpg.de

DEMOS Data Format Meeting, June 11th 2026

How to represent models?

- DEMOS: preserve deterministic and stochastic physics models in a human- and machine-readable, non-ambigous form
- “Data-format” representation, like JSON (HS3, pyhf, JSS draft):
  - Not very readable and hard to write for humans
  - Less readable and writable for LLMs
  - Trivial to read and write for computer programs
- “Source-Code”:
  - Very readable and writable for humans
  - More readable and writable for LLMs
  - Harder to read and write for computer programs

How to represent models?

- Idea: Let's have both, but common semantics and round-trip capability

How to author models?

- Small to medium-sized models:
  - Could be authored directly in a human-friendly common format
  - Problem: users may want to avoid lock-in and export to what they know - one source, but many desired targets.
- Larger models:
  - Often emitted by model-building tooling
  - Problem: many incompatible sources, one target

Can we convert models?

- Two continents: “tilde land” (Stan, (Num)Pyro, …) and “PDF/distribution-land” (RooFit, HS3, JSS, BAT.jl)
- Needs advanced term-rewriting in a representation that covers both

FlatPPL

- Covers both “tilde land” (stochastic nodes) and “PDF/distribution-land” (measure algebra)
- Designed for
  - Math-like appearance (math but non-ambigous)
  - Term-rewriting
  - Full up-front data type and size inference (great for XLA/MLIR, etc.)
- FlatPPL profiles - well-defined subsets that map to specific source-target systems

FlatPPL

- Each file is a module, flat namespace in each module (like a RooFit workspace)

FlatPPL features

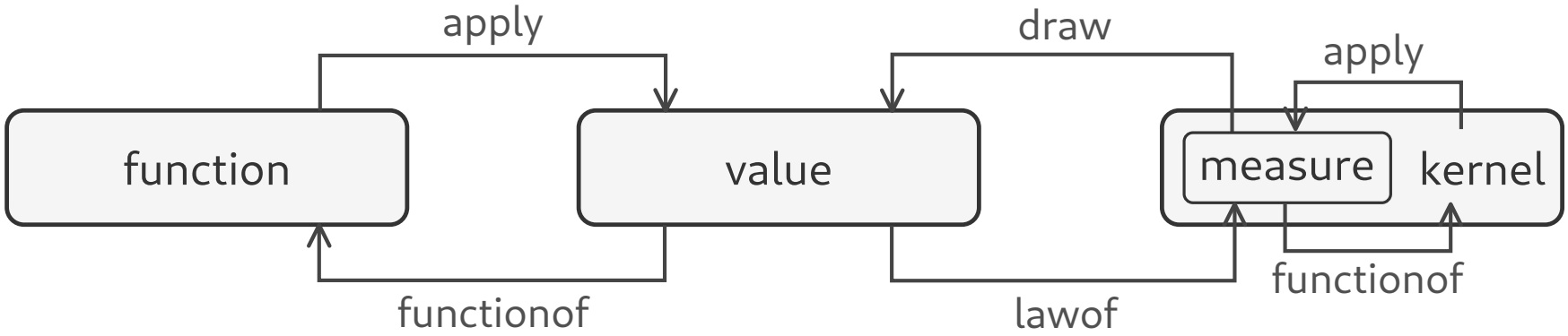
- Vectors, matrices, tensors (may be nested) with broadcasting, co-/contravariant Einstein summation, etc.
- Complex numbers
- Records (non-nested) and Tables
- Built-in and user-definable deterministic functions
- Built-in and user-definable Distributions/measures and (Markov) kernels

FlatPPL non-features

Max-Planck-Institut für Physik
(Werner-Heisenberg-Institut)

- No specific numerical data formats (float32, float64, …):
  Users and engines will control numeric precision
  (likely increasingly automatic and adaptive in the future)
- No Turing-completeness: We want predictable termination
- No file writing, URL-generation, etc.:
  want to share models with limited trust, so make it
  (near) impossible to write malicious models in FlatPPL
- No bit-level reproducibility of results:
  futile in the modern hardware landscape,
  and physics shouldn’t depend on the last bits

Term algebra



- Term algebra (likely novel, in this form) that stretches across function/kernel application, function/kernel definition, stochastic draws and distribution definition
- Hopefully the bridge between “tilde land” and “PDF/distribution-land” (to be proven in practice)

Intermediate representation (IR)

Max-Planck-Institut für Physik
(Werner-Heisenberg-Institut)

- FlatPIR: FlatPPL optionally enriched with precompile-time inferred information like stochasticity, type, array-shape, etc.
- Serializations:
  - Default: Lisp-like S-expressions (great for term-rewriting)
  - Binary: TBD, but likely FlatBuffers/Arrow
  - JSON: If required, easy to do
- FlatPPL <=> FlatPIR round trip capability (Rust tool prototype `flatppl convert`).

FlatPPL as a DEMOS model format candidate

Max-Planck-Institut für Physik
(Werner-Heisenberg-Institut)

- Pro: FlatPPL will cover all of our science domain needs
- Con: FlatPPL is new, HS3 already has community adoption
- A well-chosen FlatPPL/archive profile would be suitable for model preservation and exchange
- FlatPPL compiler would lower full FlatPPL to FlatPPL/archive
- Serialization to/from JSON and other formats very easy
- Alternatives: HS3, XS3/JSS (WIP)

FlatPPL for MaaS

Max-Planck-Institut für Physik
(Werner-Heisenberg-Institut)

- FlatPPL can cover our MaaS (model as a service) needs.
- MaaS servers wrap arbitrary systems the have things like functions and distributions in FlatPPL semantics
- FlatPPL/remote: a small FlatPPL profile with functions like `rand` and `logdensityof`
- MaaS client sends small FlatPPL/remote module as query to MaaS-server, server responds with quantities of interest
- On-the-wire protocol: FlatPIR/remote binary

Compilation/execution workflow

Max-Planck-Institut für Physik
(Werner-Heisenberg-Institut)

- Parse FlatPPL/FlatPIR
- Type/shape/stochasticity inference
- Given use case (data generation or inference) rewrite stochastic code to FlatPDL (Flat Portable Deterministic Language) profile
- Lower FlatPDL to target platform idioms (stable transformation)
- Translate lowered FlatPDL to target platform language
- Compile/run on target platform

Status

- Design in a good shape already:
  https://github.com/flatppl/flatppl-design
- Editor support (TextMate, Tree-sitter, Pygments, and more):
  https://github.com/flatppl/flatppl-grammars
- VS-Code plugin and JS FlatPPL engine prototype:
  https://github.com/flatppl/flatppl-js
- FlatPPL in Rust (very WIP):
  https://github.com/flatppl/flatppl-rust
- FlatPPL in your browser (based on flatppl-js):
  https://flatppl.github.io/flatppl-js

Next steps

- Improve JS engine, good testbed for language features
- Build out Rust ecosystem:
  - pyHF and HS3 to FlatPPL conversion (later FlatPPL to HS3)
  - Type/shape/stochasticity inference as common tooling for engines in Python and Julia
- Prototype Julia engine (likely won't need FlatPDL at first)
- Define FlatPDL profile
- Specify FlatPPL/remote-based MaaS protocol

Want to join in?