MLabs – Plutarch v2 Close-Out Report

A. Name of project and Project URL

MLabs – Plutarch v2 - 900157 - https://cardano.ideascale.com/c/idea/63765

B. Name of project manager

Ben Hart / Mark Florisson

C. Date project started

November 2022

D. Date project completed

June 2023

E. List of challenge KPIs and how the project addressed them

Overview

Plutarch is a transformative eDSL, written in Haskell, designed to optimize and simplify the development of efficient Plutus Core validators. Plutarch was created to address bottlenecks faced by dApp developers, and it is constructed to resolve limitations of the Plutus Tx compilation pipeline, offering more fine-grained control over the Plutus Core generated.

Plutarch offers several other key benefits:

- Supports a broader set of Haskell features.
- Enables more efficient usage of transaction space.
- Simplifies error messages for easier debugging.
- Allows for precise and optimal smart contract performance.

KPIs

MLabs submitted its Plutarch v2 proposal in the Fund9 Developer Ecosystem challenge. In general, the challenge aimed to provide tooling and solutions to create a better developer experience on Cardano. Suggested directions included:

- Developer productivity: IDEs, scripts to automate stuff
- Support structures
- Knowledge base and documentation

- APIs and oracles

And so on.

Although Plutarch v2 remains a WIP, it aims to be an eDSL framework for multiple backend languages, and great effort has been spent over the last year or so researching and analyzing how to best accomplish this.

F. List of project KPIs and how the project addressed them

Several approaches and objectives were realized throughout the Plutarch v2 project.

- Plutarch Nix was updated and Nix backend output added
- **pletFields** were removed due to the complications surrounding them
- fixed dichotomy in handling **PType** and **a PhsEf**
- added primitive types, coercions, and helper functions
- add the **Term** data type for encoding languages
- added **tags** to generalize **IsPtype** and **PIO**
- removed interpretIn

And more.

In general,

G. Key achievements

There are many small novel contributions, albeit the biggest one is Term. It looks like Free but without Pure. It supports reordering, and partial reinterpretation (!), without sacrificing support for linearity. It is very powerful. Other notable advancements include eDSL-level optimisations, top-level common expression elimination through hashing, avoiding redoing work through sharing thunks, supporting normal ADTs through Generics and a recursive HKD-like construction.

H. Key learnings

As development progressed, team leaders realized effect systems were analogous to languages. The early version of the project took a simple eDSL approach, however, this ended up being restrictive. Moreover, we needed to maintain types in the output of Plutarch as opposed to polymorphic output. Over time, team leaders realized that zero-cost abstractions could be realized at compile time, functioning like monomorphization in Rust.

I. Next steps for the product

There are still features that remain to be implemented or hashed out to get Plutarch v2 completely working and ergonomic enough to the extent expected by users.

- Add more utilities
- Reconfigure support for custom data types
- Implement C backend
- Reimplement a zero-cost effect system
- Possibly add support for arrow syntax
- Clean up implementation

J. Final thoughts

As mentioned, several issues remain to be overcome for Plutarch v2 to be functional. Project leaders will continue to chip away at the issues, and it is hoped that community members will also help – particularly with the C backend – as the project could benefit a wide range of projects even outside of Cardano.

K. Links

MLabs website: https://mlabs-digital.com
GH: https://github.com/Plutonomicon/plutarch-core