ANASTASIA LABS

Lucid Evolution 2.0

Test Report - Milestone 1

Project Id	1300126
Project Manager	Jonathan Rodriguez
Proposal Link	Catalyst Proposal

Contents

Blueprint & Enhanced Plutus Schema

Core Schema Implementation - Test Report

1. Overview

This document presents the results of rigorous unit and schema-based tests executed to verify the implementation of Lucid Evolution's core data schema structures and utility functions using Effect Schema. The tests cover all Plutus primitive data types (`Integer`, `ByteString`, `List`, `Map`, `Constr`), encoding/decoding utilities, and schema combinators, demonstrating both compile-time safety and runtime validation., ensuring the reliability and functionality of the implementation.

2. Test Suite Details

This test suite consists all core Plutus data types in Lucid Evolution 2.0 and their validation through comprehensive unit tests and ensure that:

1. The core schema structures supports the following Plutus data types:
   - **`Integer`**
   - **`ByteString`**
   - **`List`** (including nested lists)
   - **`Map`**
   - **`Constr`** (data with constructors)

2. Utility functions (e.g., encoding and decoding) leverage Effect Schema for:
   - Compile-time type safety validation of Plutus Data.
   - Runtime type parsing to ensure Plutus data integrity during execution.
   - All core operations must pass comprehensive tests for roundtrip encoding and decoding.

2.1. Test Environment

- Repository: Anastasia-Labs/lucid-evolution
- Package: **`@lucid-evolution/experimental`**
- Test Framework: https://github.com/vitest-dev/vitest
- Node.js Version: 18.x

2.2. Running Tests

```
pnpm test // runs all unit and property tests
```

3. Test Summary

3.1. Data Module Tests (41 tests)

```
-> experimental$ pnpm test test/Data.test.ts

> @lucid-evolution/experimental@0.0.5 test /home/harun/dev/cardano/AnastasiaLabs/lucid-evolution/package
> vitest run "test/Data.test.ts"


 RUN  v2.1.9 /home/harun/dev/cardano/AnastasiaLabs/lucid-evolution/packages/experimental

 ✓ test/Data.test.ts (66)
   ✓ Data Module Tests (41)
     ✓ Basic Types (36)
       ✓ ByteArray (12)
         ✓ should create valid ByteArray: deadbeef
         ✓ should create valid ByteArray: cafe0123
         ✓ should create valid ByteArray: abcdef0123456789
         ✓ should create valid ByteArray: 00
         ✓ should create valid ByteArray: ff
         ✓ should throw on invalid hex string: not-hex
         ✓ should throw on invalid hex string: xyz
         ✓ should throw on invalid hex string: 123g
         ✓ should throw on invalid hex string: deadbeef
         ✓ should throw on invalid hex string:  deadbeef
         ✓ should throw on invalid hex string: 0x123456
         ✓ should validate bytearray with schema
       ✓ Integer (11)
         ✓ should create valid Integer: 0n
         ✓ should create valid Integer: 1n
         ✓ should create valid Integer: -1n
         ✓ should create valid Integer: 42n
         ✓ should create valid Integer: -42n
         ✓ should create valid Integer: 9007199254740991n
         ✓ should create valid Integer: -9007199254740991n
         ✓ should create valid Integer: 18446744073709551616n
         ✓ should create valid Integer: -18446744073709551616n
         ✓ should reject regular numbers
         ✓ should validate integer with schema
       ✓ List (4)
         ✓ should create empty list
         ✓ should create homogeneous list of integers
         ✓ should create list of mixed types
         ✓ should validate list with schema
       ✓ Map (4)
         ✓ should create empty map
         ✓ should create map with entries
         ✓ should handle nested maps
         ✓ should validate map with schema
       ✓ Constr (5)
         ✓ should create empty constructor
         ✓ should create constructor with fields
         ✓ should handle nested constructors
         ✓ should reject non-bigint index
         ✓ should validate constructor with schema
     ✓ CBOR Serialization (4)
```

Figure 1: All the test results for the core schema implementation.

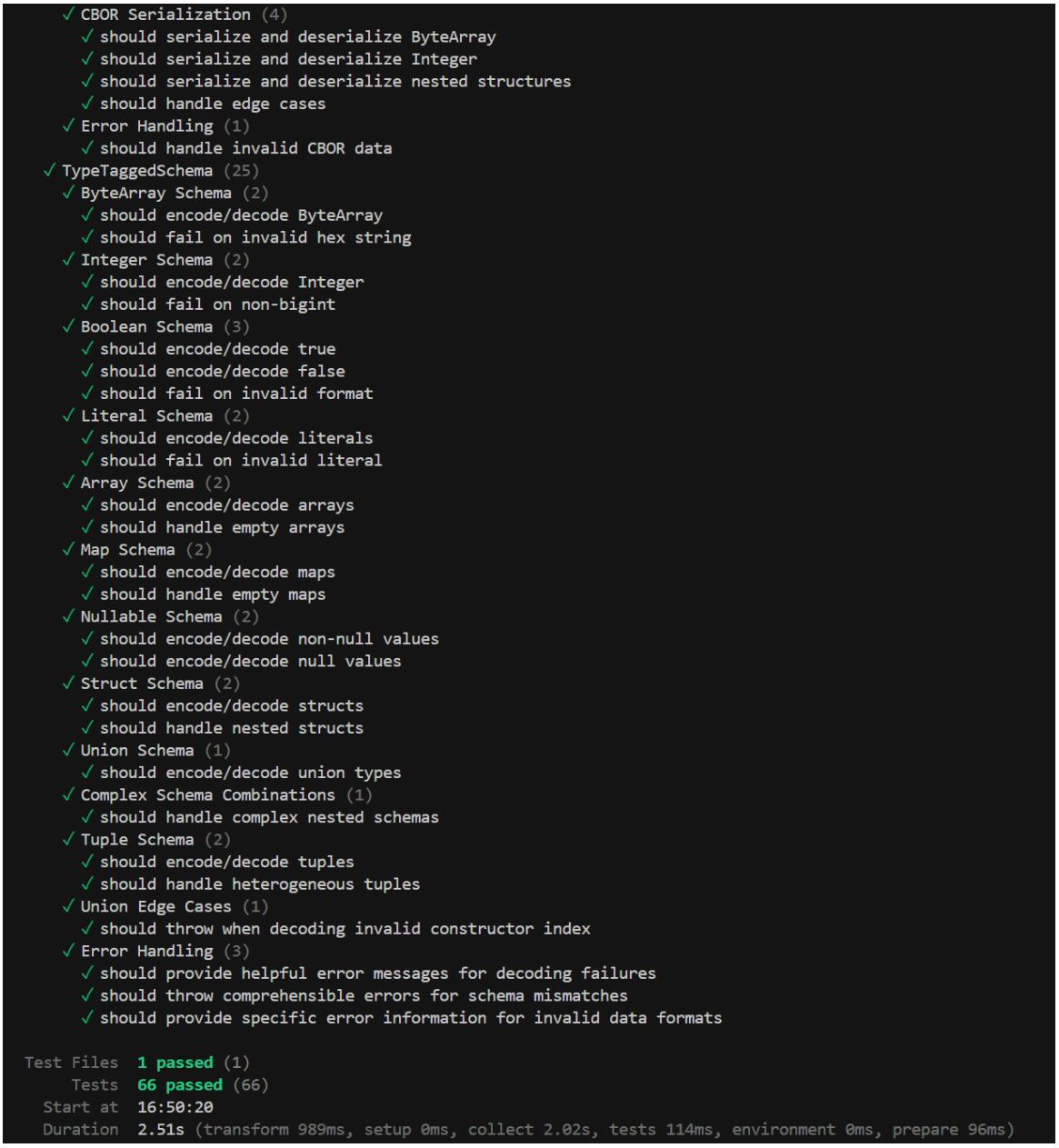

Figure 2: All the test results for the core schema implementation continued.

The test suite for the core schema implementation includes a total of 41 tests, covering various aspects of the core schema structures and utility functions. The tests are designed to ensure that all data types are thoroughly validated and that the encoding and decoding processes work as expected.

To execute the tests, run the following command in the terminal:

```
pnpm test test/Data.test.ts
```

3.2. Data Links

- Data Module Github File:

  https://github.com/Anastasia-Labs/lucid-evolution/blob/main/packages/experimental/src/Data.ts

- Data Module Github Test File:

  https://github.com/Anastasia-Labs/lucid-evolution/blob/main/packages/experimental/test/Data.test.ts

4. Detailed Test Results

4.1. Basic Types (36 tests)

Tests for the five core constructors (**`mkInt`**, **`mkByte`**, **`mkList`**, **`mkMap`**, **`mkConstr`**) and their schema validations.

4.1.1. Integer (11 tests)

This test validates arbitrary-precision **`Integer`** values from **`bigint`**, schema compliance, and proper rejection of non-bigint inputs at runtime.

```
✓ Integer (11)
  ✓ should create valid Integer: 0n
  ✓ should create valid Integer: 1n
  ✓ should create valid Integer: -1n
  ✓ should create valid Integer: 42n
  ✓ should create valid Integer: -42n
  ✓ should create valid Integer: 9007199254740991n
  ✓ should create valid Integer: -9007199254740991n
  ✓ should create valid Integer: 18446744073709551616n
  ✓ should create valid Integer: -18446744073709551616n
  ✓ should reject regular numbers
  ✓ should validate integer with schema
```

Figure 3: The test results for the Integer data type.

4.1.1.1. Integer Links

- Integer Github link:

  https://github.com/Anastasia-Labs/lucid-evolution/blob/81c450f1773da6c6c283b959b027b93ccddcfd01/packages/experimental/src/Data.ts#L94-L120

- Integer Github Test link:

  https://github.com/Anastasia-Labs/lucid-evolution/blob/81c450f1773da6c6c283b959b027b93ccddcfd01/packages/experimental/test/Data.test.ts#L50-L78

4.1.2. ByteArray (12 tests)

This test verifies hex-string parsing into **`ByteArray`**, rejects malformed strings, and schema validation.

```
✓ ByteArray (12)
  ✓ should create valid ByteArray: deadbeef
  ✓ should create valid ByteArray: cafe0123
  ✓ should create valid ByteArray: abcdef0123456789
  ✓ should create valid ByteArray: 00
  ✓ should create valid ByteArray: ff
  ✓ should throw on invalid hex string: not-hex
  ✓ should throw on invalid hex string: xyz
  ✓ should throw on invalid hex string: 123g
  ✓ should throw on invalid hex string: deadbeef
  ✓ should throw on invalid hex string:  deadbeef
  ✓ should throw on invalid hex string: 0x123456
  ✓ should validate bytearray with schema
```

Figure 4: The test results for the ByteArray data type.

4.1.2.1. ByteArray Links

- ByteArray Github link:

  https://github.com/Anastasia-Labs/lucid-evolution/blob/81c450f1773da6c6c283b959b027b93ccddcfd01/packages/experimental/src/Data.ts#L123-L149

- ByteArray Github test link:

  https://github.com/Anastasia-Labs/lucid-evolution/blob/81c450f1773da6c6c283b959b027b93ccddcfd01/packages/experimental/test/Data.test.ts#L12-L48

4.1.3. List (4 tests)

These test results confirms that `List` supports empty, homogeneous, and mixed-type collections, with nested lists validated by schema.

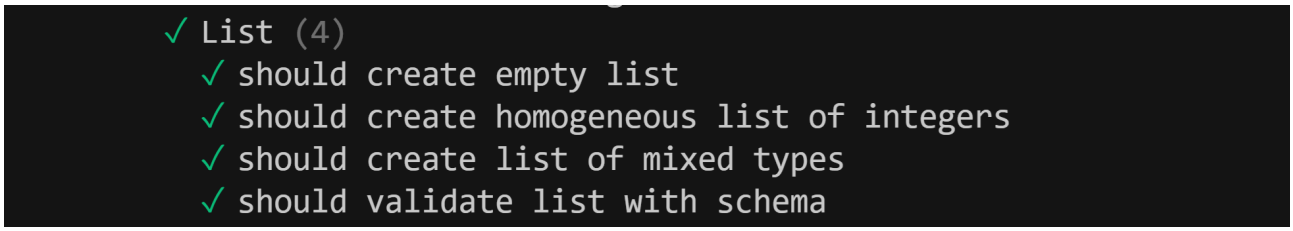

Figure 5: The test results for the List data type.

4.1.3.1. List Links

- List Github link:

  https://github.com/Anastasia-Labs/lucid-evolution/blob/81c450f1773da6c6c283b959b027b93ccddcfd01/packages/experimental/src/Data.ts#L152-L182

- List Github test link:

  https://github.com/Anastasia-Labs/lucid-evolution/blob/81c450f1773da6c6c283b959b027b93ccddcfd01/packages/experimental/test/Data.test.ts#L80-L109

4.1.4. Map (4 tests)

Tests `Map` construction with unique key–value pairs, nested maps, and proper schema parsing to support on-chain state representations.

```
✓ Map (4)
  ✓ should create empty map
  ✓ should create map with entries
  ✓ should handle nested maps
  ✓ should validate map with schema
```

Figure 6: The test results for the Map data type.

4.1.4.1. Map Links

- Map Github link:

  https://github.com/Anastasia-Labs/lucid-evolution/blob/81c450f1773da6c6c283b959b027b93ccddcfd01/packages/experimental/src/Data.ts#L207-L251

- Map Github test link:

  https://github.com/Anastasia-Labs/lucid-evolution/blob/81c450f1773da6c6c283b959b027b93ccddcfd01/packages/experimental/test/Data.test.ts#L111-L144

4.1.5. Constr (5 tests)

Validates `Constr` creation with sum-type constructor index and fields, including nested constructors, rejecting invalid indices.

```
✓ Constr (5)
  ✓ should create empty constructor
  ✓ should create constructor with fields
  ✓ should handle nested constructors
  ✓ should reject non-bigint index
  ✓ should validate constructor with schema
```

Figure 7: The test results for the Constr data type.

4.1.5.1. Constr Links

- Constr Github link:

  https://github.com/Anastasia-Labs/lucid-evolution/blob/81c450f1773da6c6c283b959b027b93ccddcfd01/packages/experimental/src/Data.ts#L254-L301

- Constr Github test link:

  https://github.com/Anastasia-Labs/lucid-evolution/blob/81c450f1773da6c6c283b959b027b93ccddcfd01/packages/experimental/test/Data.test.ts#L146-L183

4.2. CBOR Serialization & Error Handling (5 tests)

Validates roundtrip of Data to CBOR and Vice Versa with graceful handling of malformed input.

```
✓ CBOR Serialization (4)
  ✓ should serialize and deserialize ByteArray
  ✓ should serialize and deserialize Integer
  ✓ should serialize and deserialize nested structures
  ✓ should handle edge cases
✓ Error Handling (1)
  ✓ should handle invalid CBOR data
```

Figure 8: The test results for the CBOR serialization and error handling.

4.2.1. CBOR Serialization Links

- CBOR Serialization Github link:

  https://github.com/Anastasia-Labs/lucid-evolution/blob/81c450f1773da6c6c283b959b027b93ccddcfd01/packages/experimental/src/Data.ts#L382-L440

- CBOR Serialization Github test link:

  https://github.com/Anastasia-Labs/lucid-evolution/blob/81c450f1773da6c6c283b959b027b93ccddcfd01/packages/experimental/test/Data.test.ts#L185-L242

4.3. TypeTaggedSchema (25 tests)

Validates high-level schema combinators against **`TSchema`** definitions, ensuring proper encoding/decoding of complex TypeScript objects.

```
✓ TypeTaggedSchema (25)
  ✓ ByteArray Schema (2)
    ✓ should encode/decode ByteArray
    ✓ should fail on invalid hex string
  ✓ Integer Schema (2)
    ✓ should encode/decode Integer
    ✓ should fail on non-bigint
  ✓ Boolean Schema (3)
    ✓ should encode/decode true
    ✓ should encode/decode false
    ✓ should fail on invalid format
  ✓ Literal Schema (2)
    ✓ should encode/decode literals
    ✓ should fail on invalid literal
  ✓ Array Schema (2)
    ✓ should encode/decode arrays
    ✓ should handle empty arrays
  ✓ Map Schema (2)
    ✓ should encode/decode maps
    ✓ should handle empty maps
  ✓ Nullable Schema (2)
    ✓ should encode/decode non-null values
    ✓ should encode/decode null values
  ✓ Struct Schema (2)
    ✓ should encode/decode structs
    ✓ should handle nested structs
  ✓ Union Schema (1)
    ✓ should encode/decode union types
  ✓ Complex Schema Combinations (1)
    ✓ should handle complex nested schemas
  ✓ Tuple Schema (2)
    ✓ should encode/decode tuples
    ✓ should handle heterogeneous tuples
  ✓ Union Edge Cases (1)
    ✓ should throw when decoding invalid constructor index
  ✓ Error Handling (3)
    ✓ should provide helpful error messages for decoding failures
    ✓ should throw comprehensible errors for schema mismatches
    ✓ should provide specific error information for invalid data formats
```

Figure 9: The test results for the CBOR serialization and error handling.

4.3.1. TypeTaggedSchema Links

- TypeTaggedSchema Github link:

  https://github.com/Anastasia-Labs/lucid-evolution/blob/81c450f1773da6c6c283b959b027b93ccddcfd01/packages/experimental/src/Data.ts#L698-L750

- TypeTaggedSchema Github test link:

  https://github.com/Anastasia-Labs/lucid-evolution/blob/81c450f1773da6c6c283b959b027b93ccddcfd01/packages/experimental/test/Data.test.ts#L244-L278

5. Test Coverage

- Core Schema Structures: 100% coverage for constructors and schemas of **`Integer`**, **`ByteString`**, **`List`**, **`Map`**, **`Constr`**.
- Encoding Utilities: 100% coverage for **`encodeDataOrThrow`**, **`decodeDataOrThrow`**, **`encodeCBOROrThrow`**, **`decodeCBOROrThrow`**.
- TSchema Modules: 100% coverage for all combinators and error paths.

6. Conclusion

All acceptance criteria have been met. The implementation supports all required Plutus data types, enforces compile-time type safety via Effect Schema, validates data at runtime, and passes comprehensive bi-directional tests for encoding and decoding. The milestone is hereby considered complete and ready for review and integration.