

Crucible Roadmap

This document outlines potential improvements, new features, and cleanup opportunities for the Crucible test framework.

Feature Proposals

[Priority: High] Automatic Suite Discovery and Runner

Description: Add a `runAllSuites` function that automatically discovers and runs all registered test suites without requiring manual enumeration in `main`.

Rationale: Currently, projects like `protolean`, `collimator`, and `wisp` must manually list each test suite in their `main` function (see `/Users/Shared/Projects/lean-workspace/protolean/Tests/Main.lean` lines 24-76). This is error-prone and requires updating when new test files are added. The `SuiteRegistry` already tracks suites via environment extension but provides no automatic runner.

Affected Files: - `Crucible/SuiteRegistry.lean` - Add runner that iterates over `getAllSuites` - `Crucible/Core.lean` - Add `runAllSuites : IO UInt32`

Estimated Effort: Small

Dependencies: None

[Priority: High] Approximate Equality Assertions for Floating-Point

Description: Add built-in assertions for approximate float/number equality with configurable epsilon tolerance.

Rationale: Multiple projects define their own `approxEq` helpers: - `/Users/Shared/Projects/lean-workspace/tincture/TinctureTests/HarmonyTests.lean` lines 14-22 - `/Users/Shared/Projects/lean-workspace/tincture/TinctureTests/ContrastTests.lean` line 14 - `/Users/Shared/Projects/lean-workspace/afferent/Afferent/Tests/Framework.lean` lines 12-19

These are all slightly different implementations of the same concept, indicating a missing core assertion.

Affected Files: - `Crucible/Core.lean` - Add `shouldBeNear`, `shouldBeApprox` assertions

Estimated Effort: Small

Dependencies: None

[Priority: High] Test Timeout Support

Description: Add ability to specify timeouts for individual tests or test suites.

Rationale: Tests that involve I/O, network calls, or complex computations may hang. The `wisp` tests at `/Users/Shared/Projects/lean-workspace/wisp/Tests/Main.lean` demonstrate real-world need for timeouts (lines 233-253). Having timeout support built into the framework prevents hung test runs.

Affected Files: - `Crucible/Core.lean` - Add `TestCase.withTimeout` or `runTestWithTimeout` - `Crucible/Macros.lean` - Add syntax for `test "name" (timeout := 5000) := do`

Estimated Effort: Medium

Dependencies: None

[Priority: Medium] Expected Failure / Skip Test Support

Description: Add ability to mark tests as expected to fail (`xfail`) or to skip them conditionally.

Rationale: Common test framework feature for: - Documenting known bugs without breaking CI - Skipping platform-specific tests - Skipping tests that require external resources

Affected Files: - `Crucible/Core.lean` - Add `TestCase.xfail`, `TestCase.skip`, `TestCase.skipIf` - `Crucible/Macros.lean` - Add syntax like `test "name" (skip := true) := do`

Estimated Effort: Medium

Dependencies: None

[Priority: Medium] Exception Expectation Assertions

Description: Add assertions to verify that code throws expected exceptions.

Rationale: The pattern `| .error e => throw (IO.userError ...)` appears throughout test code. Need assertions like `shouldThrow`, `shouldThrowWith msg`, and `shouldNotThrow` for testing error handling code paths.

Affected Files: - `Crucible/Core.lean` - Add `shouldThrow`, `shouldThrowMatching`

Estimated Effort: Small

Dependencies: None

[Priority: Medium] Test Fixtures / Setup-Tearardown Hooks

Description: Add support for setup/teardown code that runs before/after each test or test suite.

Rationale: Many tests need shared setup (database connections, file creation, network clients). The `wisp` tests show this pattern at `/Users/Shared/Projects/lean-workspace/wisp/Tests/Main.lean` lines 883-915 with manual `globalInit/globalCleanup` calls in `main`.

Affected Files: - `Crucible/Core.lean` - Add `TestSuite` structure with `beforeAll`, `afterAll`, `beforeEach`, `afterEach` hooks - `Crucible/SuiteRegistry.lean` - Update `SuiteInfo` to include hook references

Estimated Effort: Medium

Dependencies: None

[Priority: Medium] Parallel Test Execution

Description: Add option to run tests in parallel using Lean's task system.

Rationale: Large test suites (like `ledger` with 80+ tests, `collimator` with 200+ tests) would benefit from parallel execution. Currently all tests run sequentially.

Affected Files: - `Crucible/Core.lean` - Add `runTestsParallel` with configurable concurrency

Estimated Effort: Medium

Dependencies: May need fixture support to handle shared resource initialization

[Priority: Medium] Test Filtering / Selection

Description: Add command-line arguments or configuration for running specific tests or suites.

Rationale: During development, it's useful to run only specific tests rather than the entire suite. Common patterns include name matching, tag filtering, or suite selection.

Affected Files: - `Crucible/Core.lean` - Add `runTestsFiltered` with name pattern matching - New file `Crucible/CLI.lean` - Command-line argument parsing

Estimated Effort: Medium

Dependencies: None

[Priority: Medium] Better Test Output Formatting

Description: Improve test output with colors, progress indicators, and summary statistics.

Rationale: Current output is minimal. Enhanced formatting could include: - Colored pass/fail indicators (green checkmark, red X) - Progress bar for long test runs - Timing information per test - Summary with total time, pass/fail counts by suite

Affected Files: - `Crucible/Core.lean` - Enhance `runTest` and `runTests` output - New file `Crucible/Output.lean` - Formatting utilities

Estimated Effort: Medium

Dependencies: None

[Priority: Low] Property-Based Testing Integration

Description: Add hooks or utilities for integrating with property-based testing libraries like `plausible`.

Rationale: Some projects already use `plausible` (`tincture`, `chroma`). The collimator tests at `/Users/Shared/Projects/lean-workspace/collimator/CollimatorTests/LensTests.lean` lines 334-426 show manual property testing patterns that could be formalized.

Affected Files: - New file `Crucible/Property.lean` - Property test helpers and assertions

Estimated Effort: Medium

Dependencies: Optional dependency on `plausible`

[Priority: Low] Structured Test Data Capture

Description: Add ability to capture and export test results in structured formats (JSON, JUnit XML).

Rationale: CI/CD integration often requires structured test output for reporting tools, dashboards, and trend analysis.

Affected Files: - New file `Crucible/Export.lean` - Test result serialization

Estimated Effort: Medium

Dependencies: None

[Priority: Low] Snapshot Testing

Description: Add support for snapshot testing where expected output is stored in files and compared against actual output.

Rationale: Useful for testing complex output (rendered text, serialized data) without writing complex equality checks.

Affected Files: - New file `Crucible/Snapshot.lean` - Snapshot management and comparison

Estimated Effort: Large

Dependencies: None

Code Improvements

[Priority: High] Unified Test Runner Interface

Current State: The `runTests` function returns `IO UInt32` where the exit code represents failure count. Projects must manually accumulate exit codes (see `protolean`, `collimator`, `wisp` main functions).

Proposed Change: Return a structured `TestResults` type with pass/fail counts, timing, and provide a standard `toExitCode` method.

Benefits: Cleaner API, better composability, enables richer reporting.

Affected Files: - `Crucible/Core.lean`

Estimated Effort: Small

[Priority: Medium] Improve Test Name Collision Handling

Current State: In `Crucible/Macros.lean` lines 62-77, test name collisions are handled by appending numeric suffixes. This happens silently.

Proposed Change: Add a warning when duplicate test names are detected, or make the generated name more predictable (include line number).

Benefits: Better debugging experience, clearer test identification.

Affected Files: - `Crucible/Macros.lean`

Estimated Effort: Small

[Priority: Medium] Type-Safe Test Tags

Current State: No tagging system exists for categorizing tests.

Proposed Change: Add a tag system using a custom type or string array, enabling test filtering by category (unit, integration, slow, etc.).

Benefits: Better organization, selective test runs.

Affected Files: - `Crucible/Core.lean` - Add tags to `TestCase` - `Crucible/Macros.lean` - Add tag syntax

Estimated Effort: Medium

[Priority: Low] Lazy Test Case Evaluation

Current State: All `TestCase` structures are created at module load time.

Proposed Change: Use thunks for the `run` field to defer test body construction until execution.

Benefits: Potentially faster startup for large test suites.

Affected Files: - `Crucible/Core.lean`

Estimated Effort: Small

Code Cleanup

[Priority: High] Standardize Infix Operator Documentation

Issue: The infix operators `<` and `>` defined at lines 69-70 of `Crucible/Core.lean` lack comprehensive documentation about their precedence and behavior.

Location: `/Users/Shared/Projects/lean-workspace/crucible/Crucible/Core.lean` lines 69-70

Action Required: Add detailed docstrings explaining: - What each operator does - Precedence level (currently 50) - How to type the Unicode characters - Comparison with named functions

Estimated Effort: Small

[Priority: Medium] Remove Legacy `ensureEq` Signature

Issue: The `ensureEq` function at line 20 has a non-standard parameter order (`msg`, `expected`, `actual`) compared to modern assertions (`actual`, `expected`). The comment says “legacy signature for backwards compatibility.”

Location: `/Users/Shared/Projects/lean-workspace/crucible/Crucible/Core.lean` line 19-22

Action Required: 1. Evaluate if any dependents still use this 2. Add deprecation warning 3. Eventually remove or rename to `ensureEqLegacy`

Estimated Effort: Small

[Priority: Medium] Add Module-Level Documentation

Issue: While there are file-level doc comments, the overall architecture and usage patterns are not well documented.

Location: All source files

Action Required: - Add comprehensive module docstrings - Document the relationship between `Core`, `Macros`, and `SuiteRegistry` - Add usage examples in doc comments

Estimated Effort: Small

[Priority: Low] Consolidate Error Message Format

Issue: Error messages across different assertions have slightly different formats. Some prefix with “Expected”, others with “Assertion failed:”.

Location: `Crucible/Core.lean` lines 15-66

Action Required: Standardize error message format across all assertions for consistent output.

Estimated Effort: Small

[Priority: Low] Add CLAUDE.md Project Documentation

Issue: Unlike other projects in the workspace, crucible lacks a CLAUDE.md file for AI assistant guidance.

Location: Project root

Action Required: Create CLAUDE.md documenting: - Build commands - Architecture overview - Testing approach (meta - how to test the test framework)

Estimated Effort: Small

API Enhancements

[Priority: High] Rich Comparison Assertions

Description: Add assertions that provide better diff output for complex types.

Proposed Additions: - `shouldContainAll` - Check list contains all expected elements - `shouldHaveKeys` - Check map/dict has expected keys - `shouldStartWith`, `shouldEndWith` - String prefix/suffix checks - `shouldMatchRegex` - Regular expression matching

Affected Files: - `Crucible/Core.lean`

Estimated Effort: Medium

[Priority: Medium] Assertion Context / Custom Messages

Description: Allow adding custom context messages to any assertion.

Example:

```
(actual expected) |> withContext "checking user authentication"
```

Affected Files: - `Crucible/Core.lean`

Estimated Effort: Small

[Priority: Low] Soft Assertions

Description: Add assertions that record failures but don't stop test execution, allowing multiple checks per test.

Rationale: Sometimes useful to check multiple conditions and see all failures at once.

Affected Files: - `Crucible/Core.lean` - Add `softAssert` family of functions - Possibly need a test context monad to track soft failures

Estimated Effort: Medium

Notes

Dependencies on This Project

The following projects depend on crucible: - afferent - arbor - chroma - collimator - enchiridion - ledger - legate - protolean - terminus - tincture - trellis - wisp

Any breaking changes should be carefully considered and communicated.

Lean Version

Currently targeting Lean 4.26.0. Feature additions should be compatible with this version.