Temporal Repair and Schrodinger Bridges

Persistent-world repair, compilation plans, and bridge execution traces

Simon Frost

Table of contents

Introduction

`FunctorFlow.jl` now includes the full v1 temporal surface: persistent states, trajectories, temporal repair objects, and Schrodinger-bridge summaries. This vignette shows the highest-level Julia-native path through that stack:

1. build the bundled temporal repair example;
2. inspect the repaired trajectory and bridge summary;
3. compile the example into parity-oriented semantic artifacts; and
4. execute the lowered placeholder IR to expose the temporal trace.

Setup

```
using Pkg
Pkg.activate(joinpath(@__DIR__, ".."))

using FunctorFlow
```

Build the bundled temporal repair example

`build_temporal_repair_example()` returns a self-contained example with raw states, repaired states, trajectories, a temporal block, a repair object, and a bridge object.

```
example = build_temporal_repair_example()
summary = summarize_temporal_repair_example(example)

println("Counts: ", summary["counts"])
println("Bridge summary: ", summary["bridge"])
println("Company summary: ", summary["company_summaries"][1])
```

```
Counts: Dict("temporal_repairs" => 1, "repaired_states" => 3, "repaired_trajectories" => 1, "r
Bridge summary: Dict{String, Any}("summary_metrics" => Dict("csb_sde_mean" => Dict("mse" => 0.
Company summary: Dict{String, Any}("company" => "acme", "years" => [2023, 2024, 2025], "tempor
```

The repaired 2024 state inserts a stabilizing action, so the temporal repair is not just a relabeling; it changes the intermediate state while preserving the endpoint-compatible trajectory.

```
raw = example[:raw_trajectory]
repaired = example[:repaired_trajectory]
repair = example[:temporal_repair]
bridge = example[:bridge]

println("Raw trajectory years: ", raw.years)
println("Raw state names: ", [state.name for state in raw.states])
println("Repaired state names: ", [state.name for state in repaired.states])
println("Repair objective: ", repair.repair_objective)
println("Repair map components: ", sort([component.name for component in values(repair.repair_
println("Bridge metrics: ", bridge.summary_metrics)
```

```
Raw trajectory years: [2023, 2024, 2025]
Raw state names: [:AcmeRaw2023, :AcmeRaw2024, :AcmeRaw2025]
Repaired state names: [:AcmeRepaired2023, :AcmeRepaired2024, :AcmeRepaired2025]
Repair objective: temporal_block_denoising
Repair map components: [:AcmeTemporalRepair__repair_2023, :AcmeTemporalRepair__repair_2024, :A
Bridge metrics: Dict(:csb_sde_mean => Dict(:mse => 0.12, :mae => 0.08), :conditional_flow => D
```

Compile the temporal stack

The semantic compiler treats the temporal repair and the Schrodinger bridge as first-class compiled artifacts.

```
plan = build_temporal_repair_compilation_plan(example)

compiled_subjects = [
    (artifact.subject_name, artifact.subject_kind)
    for artifact in plan.artifacts
    if artifact.subject_kind in (
        :trajectory_functor,
        :temporal_block,
        :temporal_repair,
        :temporal_schrodinger_bridge,
    )
]

println("Compiled temporal subjects:")
for subject in compiled_subjects
    println("  ", subject)
end
```

```
Compiled temporal subjects:
  (:AcmeRawTrajectory, :trajectory_functor)
  (:AcmeRepairedTrajectory, :trajectory_functor)
  (:AcmeTemporalBlock, :temporal_block)
  (:AcmeTemporalRepair, :temporal_repair)
  (:AcmeTemporalBridge, :temporal_schrodinger_bridge)
```

Lowering the plan to placeholder IR makes the temporal semantics explicit as operations such as `repair_temporal_block` and `instantiate_temporal_bridge`.

```
ir = build_temporal_repair_executable_ir(example)

println("IR instructions:")
for instruction in ir.instructions
    if instruction.opcode in (:repair_temporal_block, :instantiate_temporal_bridge)
        println("  ", instruction.name, " ⇒ ", instruction.opcode)
    end
end
```

```
IR instructions:
  AcmeTemporalRepair__temporal_repair => repair_temporal_block
  AcmeTemporalBridge__temporal_schrodinger_bridge => instantiate_temporal_bridge
```

Execute the placeholder trace

The execution trace is symbolic rather than numeric, but it exposes the exact categorical steps the temporal example lowers to.

```
executed = execute_temporal_repair_example(example)

println("Execution trace:")
for line in executed.trace
    println("  ", line)
end
```

```
Execution trace:
  AcmeRaw2024__persistent_state: declare_persistent_state inputs=() outputs=(:AcmeRaw2024,) ty
  AcmeRaw2025__persistent_state: declare_persistent_state inputs=() outputs=(:AcmeRaw2025,) ty
  AcmeRaw2023__persistent_state: declare_persistent_state inputs=() outputs=(:AcmeRaw2023,) ty
  AcmeRepaired2024__persistent_state: declare_persistent_state inputs=() outputs=(:AcmeRepaire
  AcmeRepaired2025__persistent_state: declare_persistent_state inputs=() outputs=(:AcmeRepaire
  AcmeRepaired2023__persistent_state: declare_persistent_state inputs=() outputs=(:AcmeRepaire
  AcmeRawTrajectory__trajectory: declare_trajectory_functor inputs=(:AcmeRaw2023, :AcmeRaw2024
  AcmeRepairedTrajectory__trajectory: declare_trajectory_functor inputs=(:AcmeRepaired2023, :A
  AcmeTemporalBlock__temporal_block: declare_temporal_block inputs=() outputs=(:AcmeTemporalBl
  AcmeTemporalRepair__temporal_repair: repair_temporal_block inputs=(:AcmeRawTrajectory, :Acme
  AcmeTemporalBridge__temporal_schrodinger_bridge: instantiate_temporal_bridge inputs=(:AcmeRa
```

Why this matters

The v1 temporal layer is now documented as an actual Julia workflow, not just a test surface:

- persistent states give year-indexed categorical state objects;
- temporal repair records how a corrupted trajectory is repaired; and
- Schrodinger bridges summarize the endpoint-conditioning geometry that ties raw and repaired trajectories together.

That makes the Julia port usable for both symbolic parity checks and future Lux-backed temporal modeling work.