Bisimulation Quotient

Collapsing behaviorally equivalent controllers with a coequalizer

Simon Frost

Table of contents

Setup

```
using FunctorFlow
```

`bisimulation_quotient_block` is the categorical move that turns two behavior maps into an explicit quotient. This vignette uses two different latent controller state spaces that nonetheless induce the same observable routing policy on paired states.

Build the Example

```
example = build_bisimulation_quotient_example()
diagram = example[:diagram]
println(summary(diagram))
println("Declared bisimulations: ", collect(keys(get_bisimulations(diagram))))
```

```
FunctorFlow.Diagram
Declared bisimulations: [:controller_alignment]
```

The example explicitly declares:

1. two observation coalgebras (`controller_a` and `controller_b`),
2. a bisimulation witness (`controller_alignment`), and
3. a coequalizer quotient that identifies equivalent behavior classes.

Execute the Quotient Diagram

```
result = execute_bisimulation_quotient_example(example)
println("Left behaviors: ", result.values[diagram.ports[:left_behavior].ref])
println("Right behaviors: ", result.values[diagram.ports[:right_behavior].ref])
println("Quotient codes: ", result.values[diagram.ports[:output].ref])
println("Coequalizer loss: ", result.losses[:behavioral_class_coeq_loss])
```

```
Left behaviors: Dict(:watchful_recovery => (0.0, 1.0), :acute_reroute => (1.0, 2.0), :steady_p
Right behaviors: Dict(:watchful_recovery => (0.0, 1.0), :acute_reroute => (1.0, 2.0), :steady_
Quotient codes: Dict(:watchful_recovery => 10.0, :acute_reroute => 21.0, :steady_progress => 0
Coequalizer loss: 0.0
```

The key point is that the latent states are not identical, but their observable behavior signatures agree pairwise. The coequalizer loss therefore vanishes.

Summarize the Quotient Classes

```
summary = summarize_bisimulation_quotient_example(example)
println("Quotient labels: ", summary["quotient_labels"])
println("Declared bisimulations: ", summary["declared_bisimulations"])
println("Counts: ", summary["counts"])
```

```
Quotient labels: Dict("watchful_recovery" => "watchful_recovery", "acute_reroute" => "acute_re
Declared bisimulations: ["controller_alignment"]
Counts: Dict("paired_states" => 3, "declared_bisimulations" => 1, "quotient_classes" => 3)
```

Here the quotient classes are simple numeric codes rendered back into semantic labels like `acute_reroute` and `steady_progress`. In a larger system this same pattern can be used to collapse equivalent latent world states, controller modes, or option policies before downstream planning.

Inspect the Underlying Relation

```
println("Paired latent states: ", example[:relation_pairs])
```

```
Paired latent states: Dict{Symbol, @NamedTuple{left::@NamedTuple{acuity::Float64, blocked::Boo
```

This is the practical interpretation of the categorical quotient: start from a relation witness, project into behavior, and identify states that are indistinguishable at the behavioral level.

For the broader CATAGI symbolic overview, see 18 Neurosymbolic Pipelines. For the trainable version of this pattern, see `build_bisimulation_quotient_lux_model` in 07 Lux Neural Backend.