Pre-Collapse Rescue Harness

--- add these operators alongside the existing ones in the seed ---

```
def op_warning_to_S5(membrane):
"""
Move from S2 or S3 into S5 (Pre-Collapse Overload).
"""
if membrane.cop_state not in (COPState.S2_CONSTRUCTION,
COPState.S3_STABILIZING_DRIFT):
raise ValueError(f"Warning operator only valid from S2 or S3, not {membrane.cop_state}")
# coherence drifting toward collapse, boundary permeable/failing is implied structurally
membrane.coherence = Coherence.DRIFTING
membrane.boundary = Boundary.PERMEABLE
return COPState.S5_PRE_COLLAPSE

def op_reframe_to_S6(membrane):
"""
Reframe after pre-collapse: S5 -> S6.
"""
if membrane.cop_state != COPState.S5_PRE_COLLAPSE:
raise ValueError(f"Reframe operator only valid from S5, not {membrane.cop_state}")
membrane.coherence = Coherence.STABLE
membrane.boundary = Boundary.EXPLICIT
return COPState.S6_COHERENT_REFRAME

def op_stabilize_to_S1(membrane):
"""
Stabilize after reframe: S6 -> S1 (return to stable framed inquiry).
"""
if membrane.cop_state != COPState.S6_COHERENT_REFRAME:
raise ValueError(f"Stabilize-to-S1 only valid from S6, not {membrane.cop_state}")
membrane.coherence = Coherence.STABLE
membrane.boundary = Boundary.EXPLICIT
return COPState.S1_INQUIRY
```

Wrap them as COPOperators

warning_op = COPOperator("Collapse-Guard-Warning", op_warning_to_S5, TransitionType.WARNING)
reframe_op = COPOperator("Coherent-Reframe", op_reframe_to_S6, TransitionType.REFRAME)
stabilize_op = COPOperator("Post-Reframe-Stabilize", op_stabilize_to_S1, TransitionType.STABILIZE)

--- Pre-Collapse Rescue trajectory harness ---

def run_pre_collapse_rescue():
"""
Implements the canonical Pre-Collapse Rescue pattern:

```
    S2/S3 -> S5 -> S6 -> S1/S3
    TT: Warning -> Reframe -> Stabilize/Inquire
"""
membrane = Membrane()

# Start in S2 (Stable Framed Construction) to match the pattern
membrane.cop_state = COPState.S2_CONSTRUCTION

usi_seq = USISequence()

# Step 1: Warning -> S5
u1 = membrane.step(warning_op, anchor="pre-collapse-warning")
usi_seq.append(u1)

# Step 2: Reframe -> S6
u2 = membrane.step(reframe_op, anchor="pre-collapse-reframe")
usi_seq.append(u2)

# Step 3: Stabilize -> S1
u3 = membrane.step(stabilize_op, anchor="post-reframe-stabilize")
usi_seq.append(u3)

# Print trajectory
for u in usi_seq.units:
    print(u.cop_expr)

# Validate USI and COP sequences
print("USI-Seq valid:", usi_seq.is_valid())
print("COP sequence globally valid:",
Membrane.cop_sequence_valid(membrane.history))
```

```
    return membrane, usi_seq
```

if **name** == "**main**":
# assuming the rest of the seed is already defined above
membrane, seq = run_pre_collapse_rescue()