Contents

ADR-002: Meta-Cognitive Bridge with 4×4 pymdp Matrices

Status

Accepted

Context

The original Cato design attempted to model 800+ knowledge domains directly in pymdp (Active Inference), creating intractable 800×800 transition matrices. This approach has several fatal flaws:

1. **Computational Intractability**: 800×800 matrices require $O(n^3)$ operations for belief updates
2. **Semantic Overload**: pymdp is designed for discrete state-action spaces, not semantic reasoning
3. **Mixing Concerns**: Conflates "what to think about" (semantics) with "how to think" (metacognition)

The fundamental insight is that **LLMs already handle semantic complexity**. What pymdp should control is **attention and cognitive mode**, not content.

Decision

Implement a Meta-Cognitive Bridge where pymdp operates on **4 discrete meta-cognitive states**, not 800+ domain states.

Meta-Cognitive States (Hidden States)

```
S = [CONFUSED, CONFIDENT, BORED, STAGNANT]
```

State	Description	Trigger Conditions
CONFUSED	High uncertainty, needs clarification	Contradictory info, novel domain
CONFIDENT	High certainty, ready to act	Consistent predictions, familiar domain
BORED	Low novelty, seeks stimulation	Repetitive patterns, no learning progress
STAGNANT	Stuck, needs external input	No progress despite effort

Meta-Cognitive Actions

`A = [EXPLORE, CONSOLIDATE, VERIFY, REST]`

Action	Description	Execution
EXPLORE	Seek new information	Generate curiosity questions
CONSOLIDATE	Strengthen existing knowledge	Memory consolidation, pattern reinforcement
VERIFY	Check understanding against reality	Tool grounding, external verification
REST	Reduce activity, wait for input	Lower temperature, passive mode

Observations (from LLM outputs)

`O = [HIGH_SURPRISE, LOW_SURPRISE, CONTRADICTION, CONFIRMATION]`

Architecture

```
                  LLM (Semantic Layer)
  Handles: Domain knowledge, language, reasoning, creativity


                             LLM Outputs


                  Signal Converter
  Extracts: Confidence scores, novelty signals, learning progress
  Maps to: Discrete observations [HIGH/LOW_SURPRISE, etc.]


                             Discrete Observations


                  pymdp (4×4 Controller)
  States: [CONFUSED, CONFIDENT, BORED, STAGNANT]
  Actions: [EXPLORE, CONSOLIDATE, VERIFY, REST]
  Computes: Optimal policy via Expected Free Energy
```

```
                              Action Policy


                   Action Executor
EXPLORE → Generate curiosity question, call LLM
CONSOLIDATE → Trigger memory consolidation pipeline
VERIFY → Call grounding tools (web search, code exec)
REST → Reduce activity, await user input
```

Signal Converter Implementation

```python
class SignalConverter:
    """Convert LLM outputs to discrete pymdp observations."""

    def __init__(self, nli_client, surprise_threshold: float = 0.5):
        self.nli = nli_client
        self.threshold = surprise_threshold

    def convert(
        self,
        prediction: str,
        outcome: str,
        confidence: float
    ) -> int:
        """
        Convert LLM prediction/outcome to observation index.

        Returns:
            0 = HIGH_SURPRISE (unexpected outcome)
            1 = LOW_SURPRISE (expected outcome)
            2 = CONTRADICTION (logical conflict)
            3 = CONFIRMATION (strong agreement)
        """
        # Use NLI to detect relationship
        result = self.nli.classify(prediction, outcome)

        if result.label == "CONTRADICTION":
            return 2  # CONTRADICTION
        elif result.label == "ENTAILMENT" and confidence > 0.8:
            return 3  # CONFIRMATION
        elif result.score < self.threshold:
            return 0  # HIGH_SURPRISE
        else:
            return 1  # LOW_SURPRISE
```

Transition Matrix (A)

The 4×4 transition matrix encodes how states evolve based on observations:

```
A[observation][current_state] → P(next_state)

Example for HIGH_SURPRISE observation:
  CONFUSED → stays CONFUSED (0.8)
  CONFIDENT → becomes CONFUSED (0.6)
  BORED → becomes interested (CONFIDENT) (0.5)
  STAGNANT → stays STAGNANT (0.7)
```

Consequences

Positive

- **Tractable computation**: 4×4 matrices compute in microseconds
- **Clean separation**: LLM handles semantics, pymdp handles control
- **Interpretable**: Meta-cognitive states are human-understandable
- **Scalable**: No growth with domain count

Negative

- **Signal conversion overhead**: Requires NLI call for each observation
- **Simplified model**: May miss nuanced cognitive states
- **Tuning required**: Transition matrices need empirical calibration

Implementation Notes

pymdp Configuration

```python
import pymdp

# 4 hidden states
num_states = [4]  # [CONFUSED, CONFIDENT, BORED, STAGNANT]

# 4 observations
num_obs = [4]  # [HIGH_SURPRISE, LOW_SURPRISE, CONTRADICTION, CONFIRMATION]

# 4 actions
num_controls = [4]  # [EXPLORE, CONSOLIDATE, VERIFY, REST]

# Initialize agent
agent = pymdp.Agent(
    A=likelihood_matrix,      # P(observation | state)
    B=transition_matrix,      # P(next_state | current_state, action)
    C=preference_vector,      # Preferred observations
    D=initial_state_prior,    # Initial state belief
    policy_len=3              # Planning horizon
)
```

Integration with Cato

The Meta-Cognitive Bridge runs as a background service, updating Cato's "mood" and guiding curiosity decisions without interfering with real-time user interactions.

References

- pymdp Documentation
- Active Inference: A Process Theory
- Free Energy Principle