Case-Enabled Reasoning Engine with Bayesian Representations for Unified Modeling (CEREBRUM)

Daniel Ari Friedman

Version 1.2 (2025-04-12)

Abstract

This paper introduces the Case-Enabled Reasoning Engine with Bayesian Representations for Unified Modeling (CEREBRUM). CEREBRUM integrates linguistic case systems with cognitive scientific principles to systematically describe, design, and deploy generative models in an expressive fashion. By treating models as case-bearing entities similar to declinable nouns in morphologically rich languages, CEREBRUM enables models to play multiple contextual roles through principled transformations. This approach establishes a formal linguistic-type calculus for cognitive model use, relationships, and transformations. The framework employs structures from category theory and techniques related to the Free Energy Principle to formalize model relationships across contexts. CEREBRUM addresses growing complexity in computational and cognitive modeling systems by providing structured representations of model ecosystems that align with lexical ergonomics, scientific principles, and operational processes.

Contents

Main Text **8**
Abstract . . . 8
Introduction . . . 8
Cognitive Systems Modeling . . . 8
Active Inference . . . 9
Linguistic Case Systems . . . 9
Intelligence Case Management Systems . . . 9
Towards Languages for Generative Modeling . . . 10
Conceptual Foundations: The Intersection of Four Domains . . . 10
Methods . . . 10
Formal Framework Development . . . 10
Mathematical Foundation . . . 12
Core Concept: Cognitive Models as Case-Bearing Entities . . . 12
Case Functions in Cognitive Model Systems . . . 12
A Prototype Case-Bearing Model: Homeostatic Thermostat . . . 15
Declinability of Active Inference Generative Models . . . 17
Morphological Transformation of Generative Models . . . 17
Active Inference Model Declension . . . 18
Model Workflows as Case Transformations . . . 20
Computational Linguistics, Structural Alignment, and Model Relationships . . . 20
Implementation in Intelligence Production . . . 20
Intelligence Production Workflow . . . 20
Active Inference Integration . . . 23
Formal Case Calculus . . . 23
Cross-Domain Integration Benefits . . . 23

Related Work . . . 31
Cognitive Architectures . . . 31
Category-Theoretic Cognition . . . 32
Active Inference Applications . . . 32
Linguistic Computing . . . 32
Conclusion . . . 32

Supplement 1: Mathematical Formalization **34**
1.1 Variational Free Energy Framework . . . 34
1.2 Markov Blanket Formulation . . . 34
1.3 Precision-Weighted Case Selection . . . 34
1.4 Dynamical Implementation . . . 34
1.5 Expected Free Energy Minimization . . . 35
1.6 Bayesian Model Comparison Between Cases . . . 35
1.7 Case-Specific Free Energy . . . 35
1.8 Core Linguistic Case Equations . . . 35
1.9 Precision-Weighted Mixture of Cases . . . 36
1.10 Composite Free Energy . . . 36
1.11 Conjunction of Models . . . 36
1.12 Conjunctive Mean Distribution . . . 36
1.13 Recursive Case Formulation . . . 37
1.14 Glossary of Variables . . . 37

Supplement 2: Novel Linguistic Cases **39**
2.1 Discovering and Creating New Linguistic Cases Through CEREBRUM . . . 39
2.2 Emergence of Novel Case Functions . . . 40
2.3 The Conjunctive Case [CNJ] . . . 40
2.3.1 Unique Properties of the Conjunctive Case . . . 40
2.4 The Recursive Case [REC] . . . 42
2.4.1 Unique Properties of the Recursive Case . . . 42
2.5 The Metaphorical Case [MET] . . . 44
2.5.1 Unique Properties of the Metaphorical Case . . . 44
2.6 Connections to Human Cognition and Communication . . . 46
2.7 Implications of Novel Cases for Computational Cognition . . . 46
2.8 Synergistic Combinations of Novel Cases . . . 47
2.9 The Explicative Case [EXP] . . . 50
2.9.1 Unique Properties of the Explicative Case . . . 51
2.10 The Diagnostic Case [DIA] . . . 53
2.10.1 Unique Properties of the Diagnostic Case . . . 53
2.11 The Orchestrative Case [ORC] . . . 54
2.11.1 Unique Properties of the Orchestrative Case . . . 55
2.12 The Generative Case [GEN] . . . 56
2.12.1 Unique Properties of the Generative Case . . . 57

Supplement 3: Practical Applications **59**
3.1 Intelligence Analysis and Production . . . 59
Security Applications . . . 59
Law Enforcement Case Management . . . 59
3.2 Healthcare and Clinical Applications . . . 59
Clinical Decision Support Systems . . . 59
Pharmaceutical Research . . . 60
3.3 Financial Services and Risk Management . . . 60
Fraud Detection Networks . . . 60
Investment Portfolio Management . . . 60

3.4 Autonomous Systems and Robotics . . . 61
Multi-Agent Robotic Systems . . . 61
Autonomous Vehicle Networks . . . 61
3.5 Natural Language Processing and Content Generation . . . 61
Enterprise Knowledge Management . . . 61
Content Generation Pipelines . . . 61
3.6 Scientific Research Applications . . . 62
Climate Modeling Consortiums . . . 62
Genomics and Bioinformatics . . . 62
3.7 Enterprise Decision Systems . . . 62
Supply Chain Optimization . . . 62
Resource Management in Complex Organizations . . . 62
3.8 Machine Learning and Neural Network Pipelines . . . 63
Deep Learning Workflow Orchestration . . . 63
Specific Actionable Scenarios in ML Pipelines . . . 63
3.9 Probabilistic Modeling and Bayesian Inference . . . 64
Bayesian Workflow Management . . . 64
Probabilistic Programming Applications . . . 65
Specific Actionable Scenarios in Probabilistic Modeling . . . 65
3.10 Drone Swarms and Coordinated Autonomous Systems . . . 66
Tactical Drone Swarm Organization . . . 66
Specific Actionable Scenarios for Drone Swarms . . . 67
3.11 Multi-Agent Hybrid/Augmented Systems with LLMs . . . 68
Human-AI Collaborative Workflows . . . 68
Specific Actionable Scenarios for LLM-Augmented Systems . . . 69
3.12 AI Safety and Interpretability . . . 69
Safety-Critical AI Systems . . . 69
Interpretability Frameworks . . . 70
Specific Actionable Scenarios for AI Safety . . . 71
3.13 Implementation and Integration Guidelines . . . 72
Cross-Domain Integration Principles . . . 72
Adaptation to Existing Systems . . . 72
Case Selection Decision Framework . . . 72

Supplement 4: Related Work **74**
4.1 Cognitive Architectures . . . 74
4.1.1 Traditional Cognitive Architectures . . . 74
4.1.2 Active Inference Cognitive Architectures . . . 74
4.2 Category-Theoretic Approaches to Cognition . . . 75
4.2.1 Categorical Compositional Cognition . . . 75
4.3 Linguistic Approaches to Computation . . . 75
4.3.1 Case Grammar and Computational Linguistics . . . 75
4.3.2 Morphological Computing and Categorical Linguistics . . . 76
4.4 Intelligence Production and Case Management . . . 76
4.4.1 Intelligence Analysis Frameworks . . . 76
4.4.2 Case Management Systems . . . 77
4.5 Emerging Approaches in Cognitive Modeling . . . 77
4.5.1 Agentic Intelligence Architectures . . . 77
4.5.2 Compositional Cognitive Systems . . . 77
4.6 Unique Contributions of CEREBRUM . . . 78
4.7 Future Integration Opportunities . . . 78
4.8 References . . . 79

Supplement 5: Category-Theoretic Formalization **81**

5.1 Introduction to Categorical Representations . . . 81
5.2 The Category of Case-Bearing Models . . . 81
5.3 Definition of Objects . . . 81
5.4 Definition of Morphisms . . . 81
5.5 Case Functors . . . 81
5.6 Functorial Representation of Case Transformations . . . 81
5.7 Natural Transformations Between Case Functors . . . 82
5.8 Commutative Diagrams for Case Transformations . . . 82
5.9 Base Transformation Diagrams . . . 82
5.10 Composition of Case Transformations . . . 82
5.11 Monoidal Structure and Case Composition . . . 82
5.12 Monoidal Category of Case Models . . . 82
5.13 Bifunctorial Properties . . . 83
5.14 Free Energy Minimization as Categorical Optimization . . . 83
5.15 Free Energy Functionals . . . 83
5.16 Optimization as Natural Transformation . . . 83
5.17 Kleisli Category for Bayesian Updates . . . 83
5.18 Stochastic Morphisms . . . 83
5.19 Bayesian Updates as Kleisli Morphisms . . . 83
5.20 Morphosyntactic Alignments as Adjunctions . . . 84
5.21 Adjoint Functors for Alignment Systems . . . 84
5.22 Universal Properties . . . 84
5.23 Practical Implementation Considerations . . . 84
5.24 Computational Representations . . . 84
5.25 Verification of Categorical Laws . . . 84
5.26 Conclusion: Categorical Foundations of CEREBRUM . . . 84

Supplement 6: Future Directions - Operational Roadmap 85
1. Core Framework Development . . . 85
    1.1 Theoretical Pathway (Conceptual Refinement & Extension) . . . 85
    1.2 Practical Pathway (Implementation & Tooling) . . . 85
2. Ecosystem & Community Building . . . 86
    2.1 Theoretical Pathway (Community Standards, Validation & Ethics) . . . 86
    2.2 Practical Pathway (Governance, Outreach, Education & Support) . . . 86
3. Interdisciplinary Integration & Application . . . 87
    3.1 Theoretical Pathway (Cross-Disciplinary Formalization & Modeling) . . . 87
    3.2 Practical Pathway (Validation, Case Studies, Domain-Specific Tools & Integration) . . . 88
4. Conclusion: An Operational Vision . . . 88

Supplement 7: Computational Complexity of Case Transformations 89
7.1 Introduction: Resource Scaling in Case-Based Cognitive Systems . . . 89
7.2 Active Inference Framework for Case-Based Computational Analysis . . . 89
7.3 Computational Complexity by Case Declension . . . 89
    7.3.1 Nominative Case [NOM] . . . 89
    7.3.2 Accusative Case [ACC] . . . 90
    7.3.3 Dative Case [DAT] . . . 90
    7.3.4 Genitive Case [GEN] . . . 90
    7.3.5 Instrumental Case [INS] . . . 90
    7.3.6 Locative Case [LOC] . . . 91
    7.3.7 Ablative Case [ABL] . . . 91
    7.3.8 Vocative Case [VOC] . . . 91
7.4 Comparative Resource Scaling Analysis . . . 92
7.5 Precision-Weighted Resource Allocation in Active Inference . . . 92

7.6 Resource Optimization Strategies for Case Transitions . . . 92
7.7 Theoretical Bounds on Case-Based Resource Optimization . . . 93
7.8 Case Selection as Resource Optimization Strategy . . . 93
7.8.1 Resource-Optimal Case Assignment Algorithm . . . 93
7.9 Practical Implications for Implementation . . . 94
7.10 Conclusion: Computational Complexity as Design Principle . . . 94
7.11 References . . . 94

Supplement 8: Active Inference Formulation Details 95
8.1 Generative Model Specification . . . 95
8.2 Free Energy Principle in CEREBRUM . . . 95
8.2.1 Relationship Between Free Energy and Case Transformations . . . 96
8.2.2 Mapping Between Active Inference and CEREBRUM . . . 96
8.2.3 Case-Specific Free Energy Applications . . . 97
8.3 Message Passing Schemes . . . 99
8.4 Expected Free Energy (EFE) for Case Selection . . . 99
8.5 Precision Dynamics . . . 100
8.6 Connections to POMDPs . . . 100
8.7 Neurobiological Connections and Computational Complexity . . . 101
8.7.1 Neurobiological Plausibility of Case-Based Active Inference . . . 101
8.7.2 Computational Complexity Implications . . . 102
8.8 Comparison with Other Active Inference Frameworks . . . 102
8.8.1 Comparative Free Energy Formulations . . . 102
8.8.2 Relationship to Variational Message Passing . . . 103
8.8.3 Extensions to Expected Free Energy . . . 103
8.8.4 Mathematical Advances Over Prior Work . . . 103

Supplement 9: Mathematical Foundations 105
9.1 Category Theory Foundations . . . 105
9.1.1 Category of Cases . . . 105
9.1.2 Functors and Natural Transformations . . . 105
9.2 Free Energy and Active Inference . . . 105
9.2.1 Free Energy Principle . . . 105
9.2.2 Case-Specific Free Energy . . . 106
9.3 Precision and Uncertainty . . . 107
9.3.1 Precision Matrices and Uncertainty . . . 107
9.3.2 Case-Specific Precision Allocation . . . 107
9.4 Multiple Dispatch and Case Polymorphism . . . 108
9.4.1 Formal Definition of Multiple Dispatch . . . 108
9.4.2 Case Polymorphism . . . 108
9.5 Information Geometry of Case Spaces . . . 109
9.5.1 Case Manifold . . . 109
9.5.2 Geodesics and Optimal Transformations . . . 109
9.6 Topological Data Analysis of Case Structures . . . 109
9.6.1 Persistent Homology . . . 109
9.6.2 Mapper Algorithm for Case Visualization . . . 110
9.7 Dynamical Systems Perspective . . . 110
9.7.1 Vector Fields and Flows . . . 110
9.7.2 Bifurcations and Case Transitions . . . 110
9.8 Computational Complexity . . . 110
9.8.1 Complexity of Case Operations . . . 110
9.8.2 Tractability and Approximations . . . 111
9.9 Convergence and Optimization . . . 111
9.9.1 Convergence of Case-Specific Learning . . . 111

9.9.2 Multi-Case Optimization . . . 112
9.10 Formal Correctness and Verification . . . 112
9.10.1 Type Theory for Cases . . . 112
9.10.2 Invariants and Properties . . . 112
References . . . 112

Supplement 10: Algorithmic Details & Pseudocode 114
10.1 Core `CaseModel` Representation . . . 114
10.2 Case Transformation Algorithm . . . 114
10.2.1 Parameter Mapping Function . . . 115
10.2.2 Precision Remapping Function . . . 115
10.2.3 Interface Configuration Function . . . 116
10.3 Free Energy Calculation Algorithm . . . 116
10.3.1 Nominative Case Free Energy . . . 116
10.3.2 Accusative Case Free Energy . . . 117
10.4 Case Selection Algorithm . . . 117
10.4.1 Information Gain Calculation . . . 118
10.4.2 Goal Alignment Calculation . . . 118
10.5 Multiple Dispatch Mechanism . . . 118
10.6 Novel Case Algorithms . . . 119
10.6.1 Conjunctive Case [CNJ] Algorithm . . . 119
10.6.2 Recursive Case [REC] Algorithm . . . 120
10.6.3 Metaphorical Case [MET] Algorithm . . . 120
10.6.4 Explicative Case [EXP] Algorithm . . . 121
10.6.5 Diagnostic Case [DIA] Algorithm . . . 121
10.6.6 Orchestrative Case [ORC] Algorithm . . . 122
10.6.7 Generative Case [GEN] Algorithm . . . 123
10.7 Database Operations for Case-Bearing Models . . . 124
10.7.1 Storing a Model . . . 124
10.7.2 Retrieving Models by Case . . . 124
10.7.3 Recording a Case Transformation . . . 125
10.8 Complexity Notes . . . 125

Supplement 11: Formal Definitions of Core Linguistic Cases 126
11.1 Introduction . . . 126
11.2 Nominative Case [NOM] . . . 126
11.2.1 Semantic Role . . . 126
11.2.2 Computational Role . . . 126
11.2.3 Formal Definition . . . 126
11.2.4 Expected Input/Output Signature . . . 126
11.2.5 Operational Definition . . . 127
11.2.6 Implementation Requirements . . . 127
11.3 Accusative Case [ACC] . . . 127
11.3.1 Semantic Role . . . 127
11.3.2 Computational Role . . . 127
11.3.3 Formal Definition . . . 127
11.3.4 Expected Input/Output Signature . . . 127
11.3.5 Operational Definition . . . 127
11.3.6 Implementation Requirements . . . 128
11.4 Genitive Case [GEN] . . . 128
11.4.1 Semantic Role . . . 128
11.4.2 Computational Role . . . 128
11.4.3 Formal Definition . . . 128
11.4.4 Expected Input/Output Signature . . . 128

11.4.5 Operational Definition . . . 128
11.4.6 Implementation Requirements . . . 128
11.5 Dative Case [DAT] . . . 129
11.5.1 Semantic Role . . . 129
11.5.2 Computational Role . . . 129
11.5.3 Formal Definition . . . 129
11.5.4 Expected Input/Output Signature . . . 129
11.5.5 Operational Definition . . . 129
11.5.6 Implementation Requirements . . . 129
11.6 Instrumental Case [INS] . . . 129
11.6.1 Semantic Role . . . 129
11.6.2 Computational Role . . . 129
11.6.3 Formal Definition . . . 130
11.6.4 Expected Input/Output Signature . . . 130
11.6.5 Operational Definition . . . 130
11.6.6 Implementation Requirements . . . 130
11.7 Locative Case [LOC] . . . 130
11.7.1 Semantic Role . . . 130
11.7.2 Computational Role . . . 130
11.7.3 Formal Definition . . . 130
11.7.4 Expected Input/Output Signature . . . 131
11.7.5 Operational Definition . . . 131
11.7.6 Implementation Requirements . . . 131
11.8 Ablative Case [ABL] . . . 131
11.8.1 Semantic Role . . . 131
11.8.2 Computational Role . . . 131
11.8.3 Formal Definition . . . 131
11.8.4 Expected Input/Output Signature . . . 132
11.8.5 Operational Definition . . . 132
11.8.6 Implementation Requirements . . . 132
11.9 Vocative Case [VOC] . . . 132
11.9.1 Semantic Role . . . 132
11.9.2 Computational Role . . . 132
11.9.3 Formal Definition . . . 132
11.9.4 Expected Input/Output Signature . . . 132
11.9.5 Operational Definition . . . 133
11.9.6 Implementation Requirements . . . 133
11.10 Case Relationships and Transformations . . . 133
11.10.1 Common Case Transformations . . . 133
11.10.2 Case Properties Summary Table . . . 133
11.10.3 Computational Implications of Case Assignment . . . 134

Supplement 12: Writing a Research Paper 135
12.1 Introduction . . . 135
12.2 Applying CEREBRUM Cases to Research Paper Components . . . 135
12.2.1 Mapping Components to Cases . . . 135
12.2.2 Key Benefits of Case-Based Approach . . . 135
12.3 The Research Paper Writing Process Through CEREBRUM . . . 136
12.3.1 Pre-Writing Phase . . . 136
12.3.2 Research Design Phase . . . 136
12.3.3 Data Collection and Analysis Phase . . . 136
12.3.4 Writing Phase . . . 137
12.3.5 Revision Phase . . . 137
12.4 Precision Management in Research Paper Writing . . . 137

12.4.1 Strategic Precision Shifting . . . 138
12.5 Communication Message Passing in Scientific Writing . . . 138
12.5.1 Internal Message Passing . . . 138
12.5.2 External Message Passing . . . 138
12.6 Practical CEREBRUM-Based Writing Strategies . . . 138
12.6.1 Case-Transitional Writing Approaches . . . 138
12.6.2 Writing Process Optimization . . . 139
12.7 Case-Based Solutions to Common Writing Challenges . . . 139
12.8 Conclusion . . . 139
12.9 Additional Tables for Research Paper Development . . . 140

Main Text

Case-Enabled Reasoning Engine with Bayesian Representations for Unified Modeling (CEREBRUM)

Daniel Ari Friedman

Version 1.2 (2025-04-12)

Institution: Active Inference Institute
Email: daniel@activeinference.institute
ORCID: 0000-0001-6232-9096
DOI: 10.5281/zenodo.15170907
URL: https://zenodo.org/records/15173983

Abstract

CEREBRUM implements a case-based approach to cognitive systems modeling by applying linguistic case frameworks to model management. The framework treats cognitive models as entities that can exist in different "cases"—analogous to nouns in morphologically rich languages—based on their functional roles within intelligence production workflows. This linguistic paradigm enables structured representation of model relationships and transformations through principled mathematical formulations. The immediate benefits include enhanced interoperability between models, reduced complexity in managing model ecosystems, and improved alignment between computational infrastructure and human cognitive patterns.

The complete source code for generating this paper, along with implementation resources and further open source development materials, are available at the CEREBRUM GitHub repository: https://github.com/ActiveInferenceInstitute/CEREBRUM.

Introduction

Cognitive Systems Modeling

Cognitive systems modeling approaches cognition as a complex adaptive system where cognitive processes emerge from dynamic interactions of multiple components across different scales. This perspective builds upon ecological psychology's emphasis on organism-environment coupling, recognizing cognitive processes as fundamentally situated in and shaped by environmental context.

The 4E cognition framework (embodied, embedded, enacted, and extended) provides a theoretical foundation for understanding how cognitive systems extend beyond individual agents

to include environmental structures and social interactions. In this view, cognitive models function as active participants in a broader cognitive ecosystem, adapting and evolving through interaction with other models and environmental constraints.

This systems-level perspective applies directly to intelligence production, where multiple analytical models must coordinate while maintaining sensitivity to changing operational contexts and requirements. The complex adaptive systems approach emphasizes self-organization, emergence, and adaptation—viewing cognitive processes as distributed across interacting components that collectively produce intelligent behavior through coordinated activity.

Active Inference

Active Inference represents a first-principles account of perception, learning, and decision-making based on the Free Energy Principle. Cognitive systems minimize variational free energy—a mathematical quantity representing the difference between an organism's internal model and its environment, functioning as an upper bound on surprise—through perception (updating internal models) and action (changing sensory inputs).

The framework formalizes uncertainty through entropy and precision weighting, enabling dynamic adaptive processes. Hierarchical message passing implementations express predictions as top-down flows and prediction errors as bottom-up flows, creating bidirectional inference systems that iteratively minimize surprise across model levels. Active Inference unifies cognitive operations as Bayesian model updates, providing a consistent mathematical formalism for predictive cognition as detailed in Supplement 11.

Linguistic Case Systems

Linguistic case systems represent grammatical relationships between words through morphological marking. These systems function as morphosyntactic interfaces between semantics and syntax, encoding contextualized relationship types rather than sequential ordering. This inherent relationality provides powerful abstractions for modeling complex dependencies and transformations between conceptual entities.

Core cases include: - Nominative (subject) - Accusative (object) - Dative (recipient) - Genitive (possessor) - Instrumental (tool) - Locative (location) - Ablative (origin)

Each serves distinct functional roles within sentence structures, with implementation varying across languages. Nominative-accusative systems distinguish subjects from objects, while ergative-absolutive systems group intransitive subjects with direct objects, as illustrated in Figure 9.

While English has largely lost morphological case marking, underlying case relationships persist through word order and prepositions. In "The cat chased the mouse," nominative case appears through position (subject before verb) rather than morphology. In "I gave him the book," dative case manifests through implied preposition and word order. These examples demonstrate that case relationships—encompassing semantics, semiosis, and pragmatics—remain fundamental to language structure regardless of explicit morphological marking.

Intelligence Case Management Systems

Intelligence case management systems organize investigative workflows and analytical processes in operational contexts. These systems structure information collection, analysis, evaluation, and dissemination while tracking provenance and relationships between intelligence products, as depicted in Figure 6. Modern implementations increasingly manage complex model ecosystems where analytical tools, data sources, and products interact within organizational workflows. However, current frameworks lack formal mathematical foundations

for representing model relationships, leading to ad hoc integration approaches that become unwieldy at scale. As artificial intelligence components proliferate in these systems, a more rigorous basis for model interaction becomes essential for maintaining operational coherence and analytical integrity, a gap specifically addressed by CEREBRUM.

Towards Languages for Generative Modeling

The Active Inference community has explored numerous adjectival modifications of the base framework, including:

1. **Deep Active Inference**: Incorporating deep learning architectures
2. **Affective Active Inference**: Integrating emotional and motivational factors
3. **Branching-Time Active Inference**: Modeling temporal contingencies and future planning
4. **Quantum Active Inference**: Applying quantum probability formulations
5. **Mortal Active Inference**: Addressing finite time horizons and existential risk
6. **Structured Active Inference**: Incorporating structured representations and symbolic reasoning

Each adjectival-prefixed variant emphasizes specific architectural aspects or extensions of the core formalism. CEREBRUM expands this paradigm by focusing on declensional semantics rather than adjectival modifications, as detailed in Supplement 2.

This approach emphasizes declensional aspects of generative models as noun-like entities, separate from adjectival qualification. This aligns with category theoretic approaches to linguistics, where morphisms between objects formalize grammatical relationships and transformations. By applying formal case grammar to generative models, CEREBRUM extends structured modeling approaches to ecosystems of shared intelligence while preserving underlying semantics that are partitioned, flexible, variational, composable, interfacial, interactive, empirical, applicable, and communicable.

Conceptual Foundations: The Intersection of Four Domains

CEREBRUM integrates four key domains to create a unified framework for model management, as illustrated in Figure 1:

1. **Cognitive Systems Modeling**: Provides the entities that take on case relationships
2. **Active Inference**: Supplies the predictive processing mechanics driving case transformations
3. **Linguistic Case Systems**: Furnishes the grammatical metaphor for model relationships
4. **Intelligence Production**: Establishes the practical application context and workflows

Together, these domains form a coherent framework where linguistic principles structure model relationships, cognitive systems supply the components, active inference formalizes transformations, and intelligence production provides the application context, as further elaborated in Supplements 7 and 11.

Methods

Formal Framework Development

The CEREBRUM framework emerged from a synthesis of linguistic theory, cognitive science, category theory, and operations research. Key methodological approaches included:

1. **Linguistic Formalization**: Adapting morphosyntactic case theory into computational representations through abstract algebraic structures. This process formalized case re-

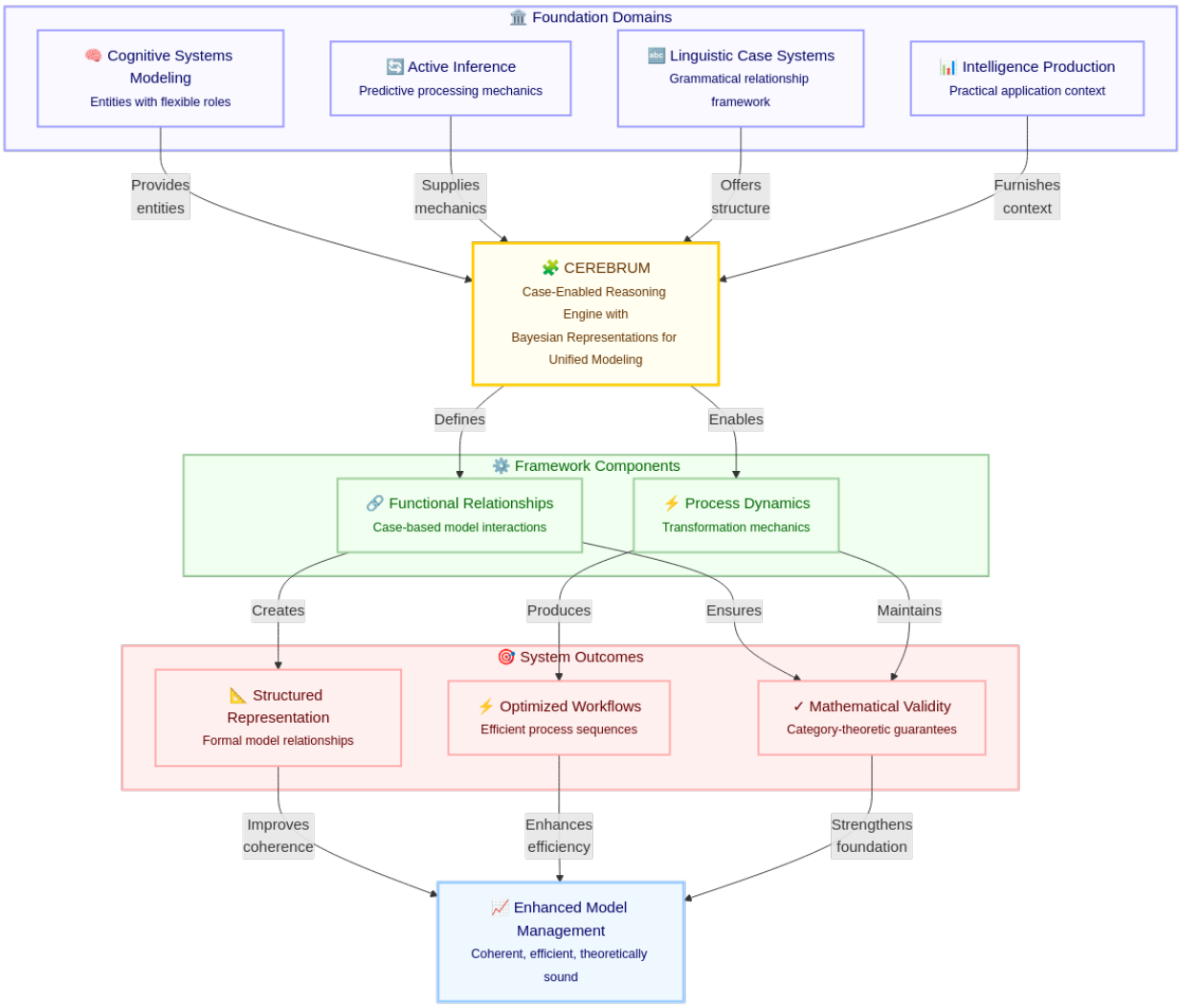


Figure 1: Figure 1. Foundation Domains of CEREBRUM. This diagram illustrates the conceptual architecture of the CEREBRUM framework, showing how four distinct domains converge to create a unified approach to model management. Cognitive Systems Modeling provides the entities that assume case relationships; Active Inference supplies the predictive processing mechanics driving case transformations; Linguistic Case Systems offer the grammatical framework for model relationships; and Intelligence Production provides the practical application context. These foundation domains integrate within CEREBRUM to produce framework components (functional relationships and process dynamics), which in turn yield system outcomes (structured representation, optimized workflows, and mathematical validity). The ultimate output is enhanced model management with improved coherence, efficiency, and theoretical soundness. This integration creates a framework that bridges theoretical linguistics, cognitive science, and practical intelligence applications.

lationships as mathematical operators that transform model properties while preserving essential characteristics.

2. **Category-Theoretic Mapping**: Implementing category theory to formalize morphisms between case states as functorial transformations. This approach enabled rigorous tracking of identity preservation across transformations while verifying compositional consistency, as formalized in Figure 7.
3. **Algorithmic Implementation**: Developing algorithmic specifications for case transformations compliant with the Free Energy Principle. These algorithms define concrete computational procedures for implementing case transitions in operational systems.
4. **Variational Methods**: Applying variational free energy calculations to optimize model inference and structural transformations. This connects case transitions to information-theoretic optimization principles, ensuring computational efficiency, as detailed in Supplement 11.

Mathematical Foundation

The mathematical foundation of CEREBRUM builds on formalizations of case transformations using category theory and variational inference. Case transformations operate as morphisms in a category where objects represent models with specific case assignments. The framework employs metrics including Kullback-Leibler divergence, Fisher information, and Lyapunov functions to quantify transformation efficacy and system stability. This approach provides both theoretical guarantees of compositional consistency and practical optimization methods for computational implementation, with formal definitions provided in Supplement 9.

Core Concept: Cognitive Models as Case-Bearing Entities

CEREBRUM's central innovation is the formalization of cognitive models as case-bearing entities that transform systematically based on their functional relationships. Just as nouns in morphologically rich languages take different forms based on grammatical function, cognitive models in CEREBRUM exist in different "states" or "cases" depending on their relationships to other models within the system. Figure 2 illustrates this linguistic parallel.

The core framework organizes cognitive models according to their case relationships, as shown in Figure 3, which maps primary case assignments and their functional roles. This structured approach enables precise transformations between model states while maintaining identity persistence across functional transitions.

The implementation of this core concept extends across multiple domains and applications, with specific mathematical formulations provided in Supplement 11 and practical applications demonstrated in Supplement 3. The case framework not only formalizes existing model relationships but also opens new possibilities for model composition and transformation that align with natural language structures, creating a systematic foundation for complex model ecosystems.

Case Functions in Cognitive Model Systems

Table 1: Case Functions in Cognitive Model Systems

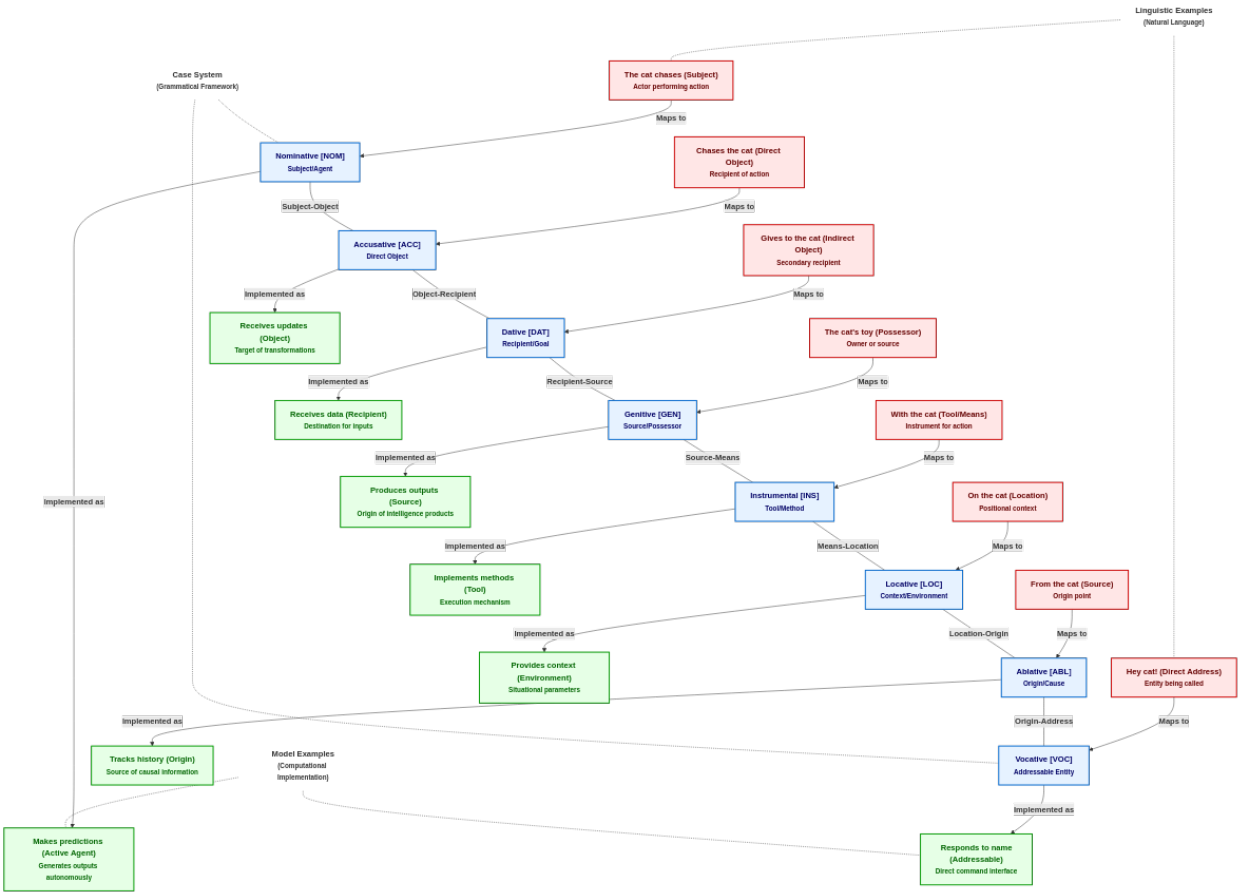


Figure 2: Figure 2. Case Relationships - Model and Linguistic Parallels. This figure illustrates the direct mapping between linguistic case systems and cognitive model relationships in CEREBRUM. The top row presents everyday linguistic examples of each case in English (though English largely expresses cases through word order and prepositions rather than morphological markers). The middle row shows the eight fundamental cases with their standard abbreviations. The bottom row demonstrates how these same case relationships apply to cognitive models within the CEREBRUM framework. For example, just as "the cat" functions as the subject (Nominative case) in language, a model in Nominative case functions as an active agent making predictions. This systematic parallel between linguistic structure and model relationships provides a principled foundation for understanding how models can assume different functional roles while maintaining their core identity, similar to how a noun retains its meaning while its form changes according to its grammatical function.

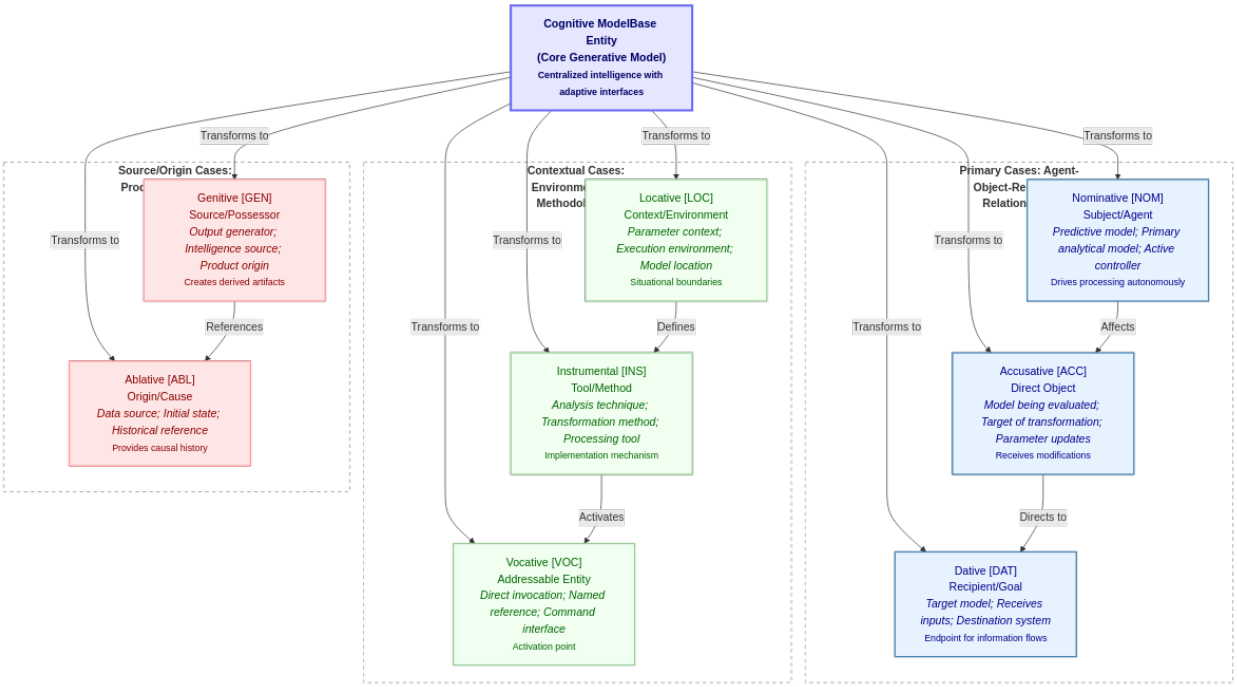


Figure 3: Figure 3. Cognitive Model Case Framework. This diagram illustrates how a single core generative model can assume different functional roles through case assignments. At the center lies the core cognitive model entity, which can be transformed into eight distinct case forms, each serving a specific function in the model ecosystem. The cases are organized into three functional groups: Primary Cases (Nominative, Accusative, Dative) handle the main agent-object-recipient relationships in model processing; Contextual Cases (Locative, Instrumental, Vocative) provide environmental, methodological, and interface functions; and Source Cases (Genitive, Ablative) manage output generation and historical attribution. Each case modifies the model's behavior, parameter access patterns, and computational interfaces while maintaining its fundamental identity. For example, when in Nominative case, the model functions as an active agent generating predictions; in Accusative case, it becomes the object of transformations; and in Genitive case, it serves as a source of outputs. This framework enables flexible, context-appropriate model behavior while preserving a coherent identity across transformations.

Abbr	Case	Function in CEREBRUM	Example Usage
[NOM]	**Nominative**	Model as active agent; produces predictions and exerts causal influence on other models	Model X generates predictions about data distributions; controls downstream processing
[ACC]	**Accusative**	Model as object of process; receives transformations and updates from other processes	Process optimizes Model X's parameters; quality assessment evaluates Model X
[GEN]	**Genitive**	Model as source/possessor; generates outputs, products, and derived models	System produces output from Model X; intelligence products derive from Model X
[DAT]	**Dative**	Model as recipient; receives and processes incoming data flows	System feeds data into Model X; Model X processes information from external sources
[INS]	**Instrumental**	Model as method/tool; implements analytical operations and procedures	System performs analysis via Model X; Model X executes analytical procedures
[LOC]	**Locative**	Model as context; establishes environmental constraints and parameters	System operates within Model X's parameters; environment exists as modeled by X
[ABL]	**Ablative**	Model as origin/cause; defines historical conditions and causal precursors	Insights derive from Model X; causal attributions trace to Model X
[VOC]	**Vocative**	Model as addressable entity; maintains callable interface with name activation	System activates Model X directly; documentation references Model X explicitly

In intelligence production systems, these case relationships fulfill distinct functional roles: nominative models drive analytical processes; accusative models receive quality assessments; genitive models generate documentation; dative models process intelligence data; instrumental models provide methodological frameworks; locative models establish situational boundaries; ablative models represent analytical origins; and vocative models serve as directly addressable interfaces. Together, these case relationships create a structured framework for intelligence workflows, as illustrated in Figure 4.

A Prototype Case-Bearing Model: Homeostatic Thermostat

Consider a cognitive model of a homeostatic thermostat that perceives room temperature and regulates it through connected heating and cooling systems:

- **Nominative [NOM]**: The thermostat actively generates temperature predictions and dispatches control signals, functioning as the primary agent in temperature regulation.
- **Accusative [ACC]**: The model becomes the object of optimization, with parameters updated based on prediction errors between expected and actual temperature readings.

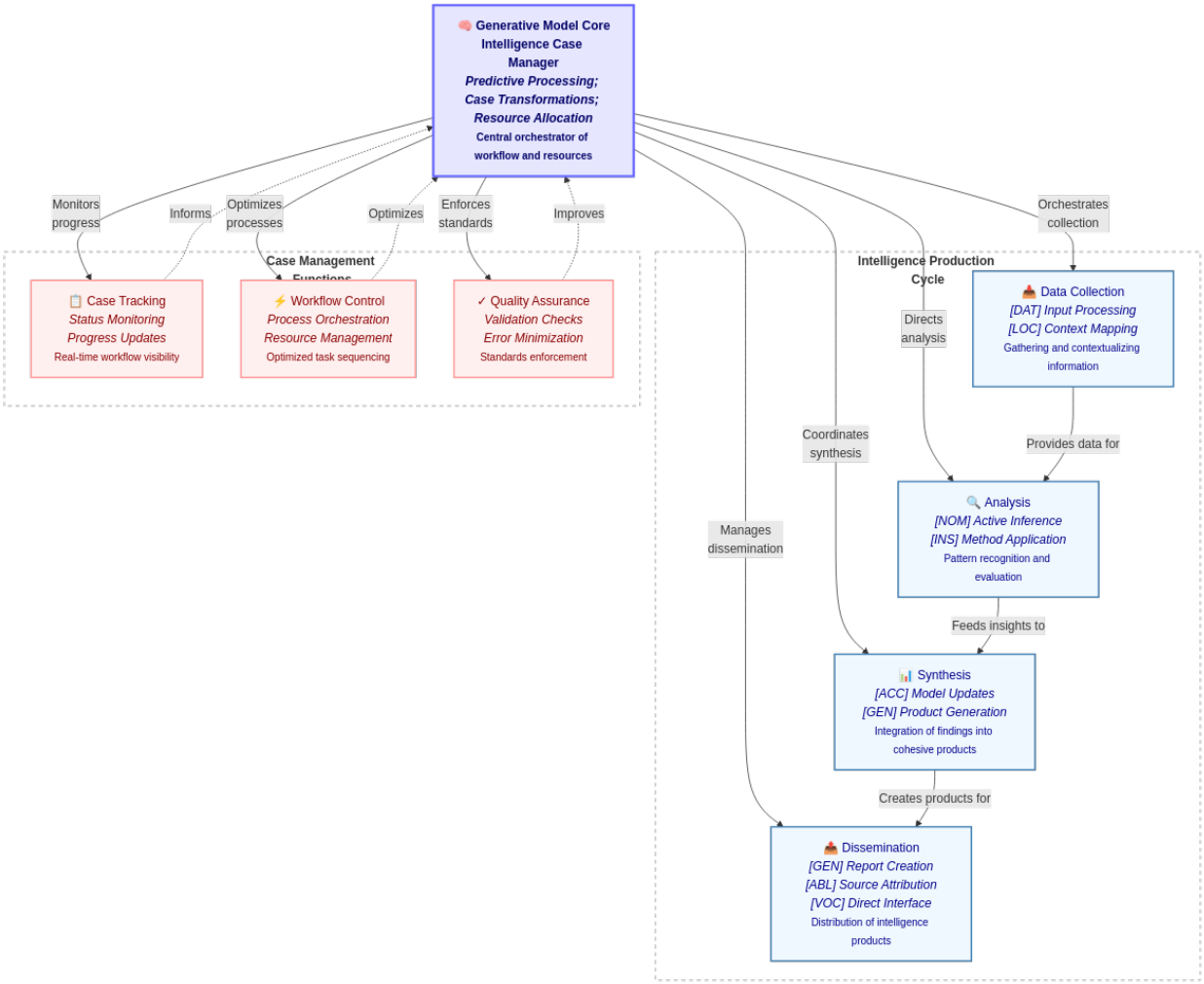


Figure 4: Figure 4. Generative Model Integration in Intelligence Case Management. This diagram illustrates how CEREBRUM's generative model core orchestrates both intelligence production and case management functions through case-specific transformations. At the center lies the core generative model that serves as the intelligence case manager, handling predictive processing, case transformations, and resource allocation. The intelligence production cycle (top) consists of four main components: Data Collection (utilizing models in Dative [DAT] and Locative [LOC] cases for input processing and context mapping); Analysis (employing models in Nominative [NOM] and Instrumental [INS] cases for active inference and method application); Synthesis (using models in Accusative [ACC] and Genitive [GEN] cases for model updates and product generation); and Dissemination (leveraging models in Genitive [GEN], Ablative [ABL], and Vocative [VOC] cases for report creation, source attribution, and direct interface). The case management functions (bottom) include Case Tracking for status monitoring, Workflow Control for process orchestration, and Quality Assurance for validation checks and error minimization. This architecture demonstrates how the assignment of specific cases to models enables dynamic role transitions as intelligence products move through the workflow, creating a flexible yet structured approach to intelligence production and management.

- **Dative [DAT]**: The thermostat receives environmental temperature data streams and occupant comfort preferences as inputs.
- **Genitive [GEN]**: The model transforms to generate temperature regulation reports and system performance analytics.
- **Instrumental [INS]**: The thermostat functions as a computational tool implementing control algorithms for other systems requiring temperature management.
- **Locative [LOC]**: The model reconfigures to represent the contextual environment, modeling building thermal properties.
- **Ablative [ABL]**: The thermostat functions as the origin of historical temperature data and control decisions, providing causal explanations for current thermal conditions.

This single cognitive model thus assumes different functional roles while maintaining its core identity as a thermostat. Supplement 3 provides additional examples of case-bearing models in various application domains.

Declinability of Active Inference Generative Models

At the core of CEREBRUM lies the concept of **declinability**—the capacity for generative models to assume different morphological and functional roles through case transformations, mirroring the declension patterns of nouns in morphologically rich languages. Unlike traditional approaches where models maintain fixed roles, CEREBRUM treats cognitive models as flexible entities capable of morphological adaptation to different operational contexts, with formal mathematical definitions provided in Supplement 11.

Morphological Transformation of Generative Models

When an active inference generative model undergoes case transformation, it experiences systematic changes including: 1. **Functional Interfaces**: Input/output specifications adapt to match case role requirements 2. **Parameter Access Patterns**: Parameter exposure or constraint patterns shift based on case 3. **Prior Distributions**: Different cases employ different prior constraints on parameter values 4. **Update Dynamics**: State update mechanisms vary by case role 5. **Computational Resources**: Different cases receive different precision-weighted computational allocations

These transformational properties are summarized in Table 2:

Table 2: Transformational Properties of Active Inference Generative Models Under Case Declensions

Case	Parametric Changes	Interface Transformations	Precision Weighting
[NOM]	Parameters configured for prediction generation	Outputs predictions; exposes forward inference pathways	Highest precision on likelihood mapping
[ACC]	Parameters configured for learning and adaptation	Receives transformations; exposes update interfaces	Highest precision on parameter updates
[DAT]	Parameters configured for input processing	Receives data flows; exposes input processing interfaces	Highest precision on incoming data
[GEN]	Parameters configured for output generation	Generates products; prioritizes output interfaces	Highest precision on generated outputs

Case	Parametric Changes	Interface Transformations	Precision Weighting
[INS]	Parameters configured for methodological operations	Implements processes; exposes computational interfaces	Highest precision on procedural execution
[LOC]	Parameters configured for contextual representation	Provides environmental constraints; exposes contextual interfaces	Highest precision on contextual representation
[ABL]	Parameters configured for causal attribution	Functions as information source; exposes historical data	Highest precision on historical data
[VOC]	Parameters configured for identification and response	Maintains addressable interfaces; exposes command channels	Highest precision on identification cues

Active Inference Model Declension

Consider a perception-oriented generative model M with parameters θ, internal states s, and observational distribution p(o|s,θ). When declined across cases, this single model transforms as follows:

- **M[NOM]**: Actively generates predictions by sampling from p(o|s,θ), with all parameters fully accessible
- **M[ACC]**: Becomes the target of updates, with parameter gradients calculated from prediction errors
- **M[DAT]**: Configured to receive data flows, with specific input interfaces activated
- **M[GEN]**: Optimized to generate outputs, with output interfaces prioritized
- **M[INS]**: Functions as a computational method, exposing algorithmic interfaces
- **M[LOC]**: Provides contextual constraints for other models, with environmental parameters exposed
- **M[ABL]**: Serves as an information source, with historical data accessible
- **M[VOC]**: Functions as an addressable entity responding to direct invocation, with naming parameters activated

The Vocative case [VOC] optimizes models for name-based recognition and command reception, particularly relevant in synthetic intelligence environments where models must be selectively activated. Vocative models maintain specialized interfaces for handling direct commands, documentation references, and initialization requests—similar to how "Hey Siri" or "OK Google" activation phrases function for digital assistants. This creates a natural bridge between human language interaction and model orchestration.

This systematic pattern of transformations constitutes a complete "declension paradigm" for cognitive models, using precision-modulation to fulfill diverse functional roles while maintaining core identity. Figure 5 illustrates the workflow of case transformations.

The implications of this declension paradigm extend beyond individual models to entire model ecosystems, as further explored in Supplements 2 and 7.

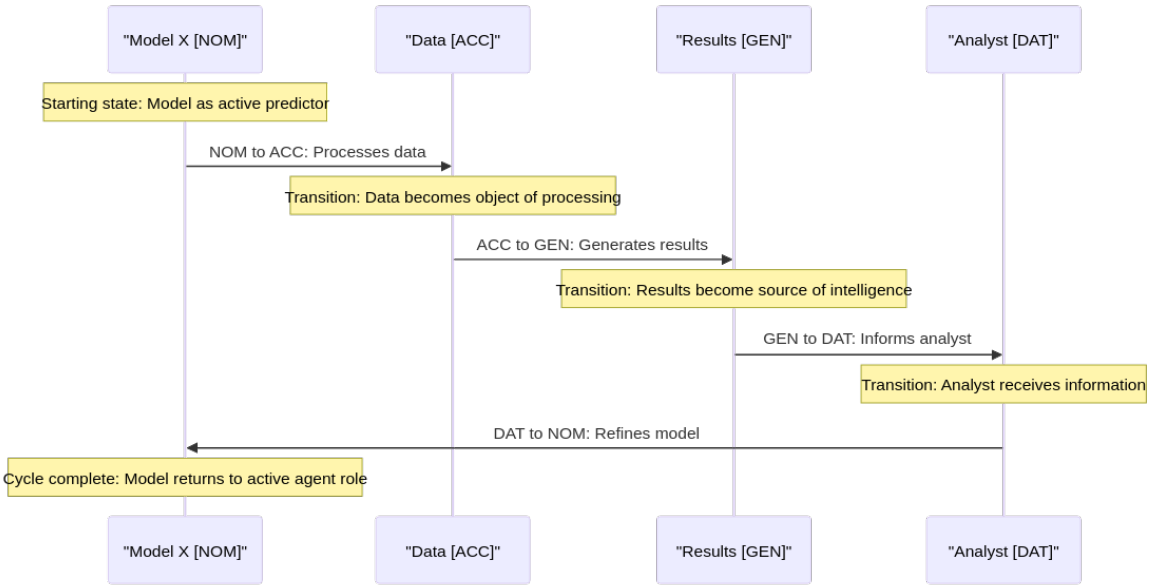


Figure 5: Figure 5. Model Workflows as Case Transformations - Sequence Diagram. This diagram illustrates the cyclical nature of case transformations in an intelligence workflow. The sequence begins with Model X in Nominative case [NOM] functioning as an active agent that processes data (in Accusative case [ACC] as the direct object of the operation). The data then transforms to Genitive case [GEN] as it generates results, becoming a source of information. These results move to Dative case [DAT] as they inform the analyst, who serves as the recipient. The cycle completes when the analyst refines the model, transforming from Dative back to Nominative case. Each arrow represents not just a data flow but a functional case transformation, where the entity changes its operational role while maintaining its core identity. This diagram demonstrates how CEREBRUM enables coherent workflows through systematic case transitions, creating a mathematically principled cycle of intelligence production where each component assumes the appropriate functional role at each stage of the process.

Model Workflows as Case Transformations

Computational Linguistics, Structural Alignment, and Model Relationships

CEREBRUM implements multiple alignment systems for model relationships, mirroring linguistic morphosyntactic structures. These alignment patterns determine how models interact and transform based on their functional roles.

Figures 9 and 10 provide complementary perspectives on alignment patterns:

- Figure 9 illustrates theoretical alignment patterns derived from linguistic theory (nominative-accusative, ergative-absolutive, and tripartite), showing the conceptual organization of models based on their case relationships.
- Figure 10 demonstrates practical computational implementation of these patterns, including specific resource allocation strategies, message passing protocols, and transformation efficiency considerations.

Implementation in Intelligence Production

CEREBRUM integrates case transformations into intelligence production workflows through several interconnected frameworks, as detailed in Supplement 3:

- Figure 6 provides the operational overview of intelligence production with case-bearing models.
- Figures 7 and 8 illustrate the category-theoretic foundations that formalize these transformations from complementary mathematical perspectives.
- Figures 11 and 12 visualize alternative state-based transitions of models between cases during intelligence lifecycles.

Intelligence Production Workflow

The CEREBRUM-managed intelligence production process follows a structured sequence of case transformations:

1. **Data Collection**: Models in instrumental case [INS] function as data collection tools, implementing specific methods for information gathering.
2. **Preprocessing**: Models transition to nominative case [NOM], becoming active agents that clean, normalize, and prepare data for analysis.
3. **Analysis**: Models transform to locative case [LOC], providing essential contextual understanding and defining environmental parameters that shape analytical processes.
4. **Integration**: Models assume genitive case [GEN], generating intelligence products by synthesizing information from multiple sources and previous stages.
5. **Evaluation**: Products undergo assessment by models in accusative case [ACC], which evaluate quality, accuracy, and relevance, identifying improvement areas.
6. **Refinement**: Models in dative case [DAT] receive feedback from evaluation and implement necessary adjustments to intelligence products or underlying analyses.
7. **Deployment**: Models return to nominative case [NOM] for active implementation or dissemination of refined intelligence solutions.

This systematic workflow demonstrates how case transformations enable models to maintain core identity while adapting to different functional requirements throughout the intelligence

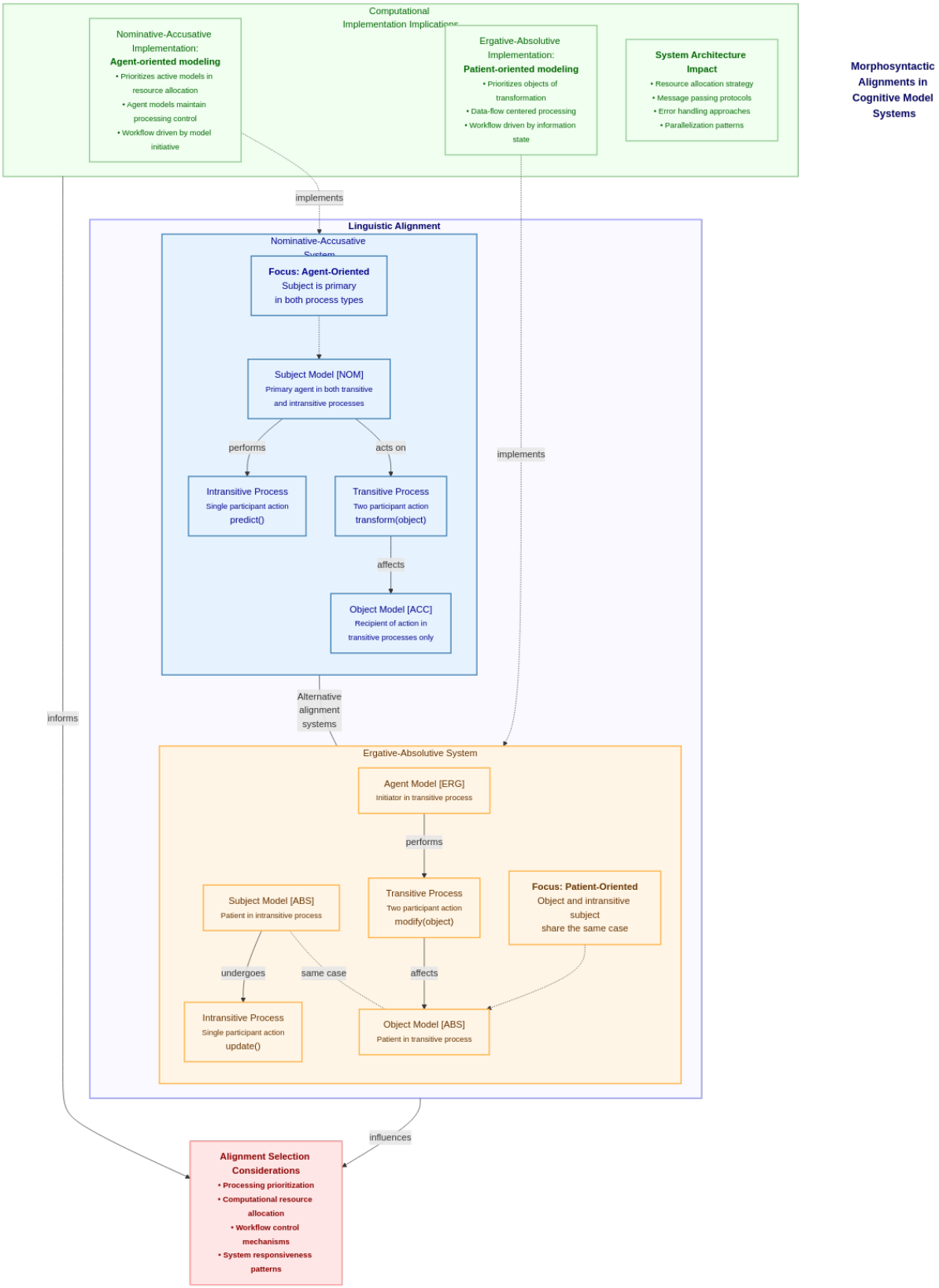


Figure 6: Figure 6. Morphosyntactic Alignments in Model Relationships. This diagram illustrates two fundamental alignment patterns that can be applied to model relationships in CEREBRUM, derived from linguistic morphosyntactic structures. The Nominative-Accusative alignment (left) groups subject models in both intransitive and transitive processes, treating them as agents regardless of process type, while differentiating object models. This pattern prioritizes agency and control flow, making it ideal for agent-centric workflows where model initiative drives processing. In contrast, the Ergative-Absolutive alignment (right) groups

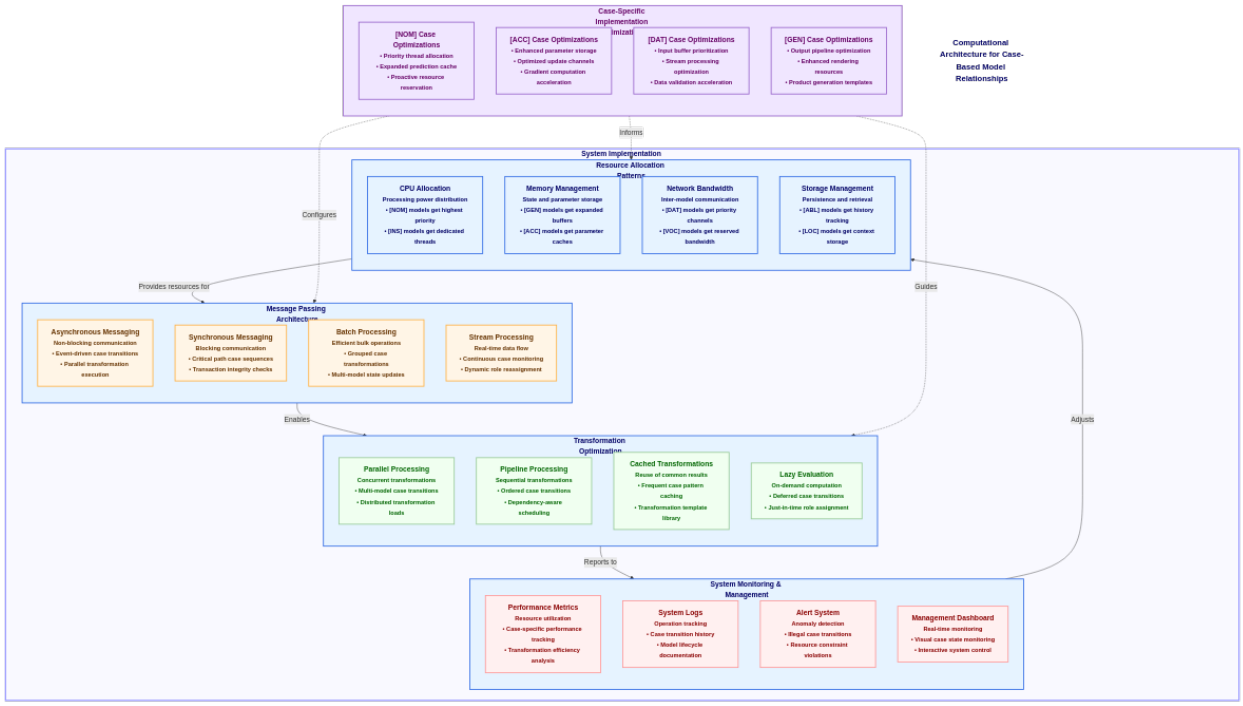


Figure 7: Figure 7. Computational Implementation of Model Relationships. This diagram illustrates the practical computational architecture required to implement CEREBRUM's case-based model relationships in real-world systems. The implementation encompasses four interconnected components that translate theoretical case transformations into efficient computational processes. Resource Allocation Patterns determine how computational resources (CPU, memory, network bandwidth, and storage) are distributed to models based on their case assignments, with nominative [NOM] models typically receiving higher processing priority while instrumental [INS] models optimize for method execution. The Message Passing Architecture defines communication protocols between case-bearing models, supporting both synchronous and asynchronous interactions, batch processing for efficiency, and stream processing for real-time applications. Transformation Optimization focuses on the efficient execution of case transformations through parallel processing, pipeline optimization, caching of common transformation patterns, and lazy evaluation for on-demand computation. System Monitoring provides observability through performance metrics, logs, alerts, and dashboards, creating a feedback loop to the resource allocation system. The Case-Specific Implementation Optimizations section highlights specialized optimizations for different cases, ensuring that each model role is computationally supported in the most efficient manner. This comprehensive implementation framework ensures that the theoretical foundations of CEREBRUM translate into scalable, efficient computational systems that can manage complex model ecosystems while maintaining the principled case relationships and transformations that define the framework.

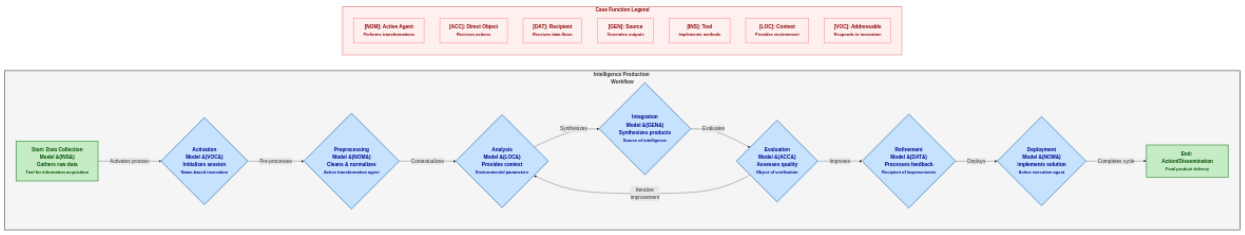


Figure 8: Figure 8. Intelligence Production Workflow with Case-Bearing Models. This flowchart depicts the intelligence production cycle from data collection to dissemination, highlighting how models in different case roles support specific stages of the workflow. The process begins with data collection using a model in Instrumental case [INS] functioning as a tool for gathering information. System activation occurs through a model in Vocative case [VOC], which serves as an addressable interface. The workflow continues with preprocessing performed by a model in Nominative case [NOM] acting as the primary agent, followed by analysis with a model in Locative case [LOC] providing contextual environment. Integration utilizes a model in Genitive case [GEN] as the source of synthesized products, while evaluation employs a model in Accusative case [ACC] as the object of quality assessment. Refinement occurs through a model in Dative case [DAT] receiving feedback, and deployment returns to a model in Nominative case [NOM] for active implementation. The diagram also shows a critical feedback loop from evaluation back to analysis, enabling iterative improvement. Each case assignment optimizes the model for its specific function in the workflow while maintaining systematic transitions between stages.

lifecycle. Each case assignment optimizes specific aspects of model behavior—from data collection and processing to product generation and quality assessment—creating a structured approach to managing intelligence workflows. Supplement 11 provides formal mathematical descriptions of these transformations, while Supplement 7 analyzes their computational complexity.

Active Inference Integration

CEREBRUM integrates with active inference principles by framing case transformations as predictive processes operating within a free energy minimization framework. This conceptual alignment is illustrated in Figure 13, which depicts case transitions driven by prediction error minimization. The specific message passing rules governing these transformations under active inference are detailed in Figure 14 and formally defined in Supplement 11.

Formal Case Calculus

The interactions and transformations between case-bearing models in CEREBRUM adhere to a formal calculus derived from grammatical case systems. This calculus defines the permissible transitions and combinatorial rules for models based on their assigned cases, as formally presented in Figure 15 and mathematically elaborated in Supplement 9.

Cross-Domain Integration Benefits

CEREBRUM's strength lies in its synthesis of concepts from four foundational domains: Linguistic Case Systems, Cognitive Systems Modeling, Active Inference, and Intelligence Production. The benefits derived from this integration are summarized in Table 4.

Table 4: Cross-Domain Integration Benefits in CEREBRUM Framework

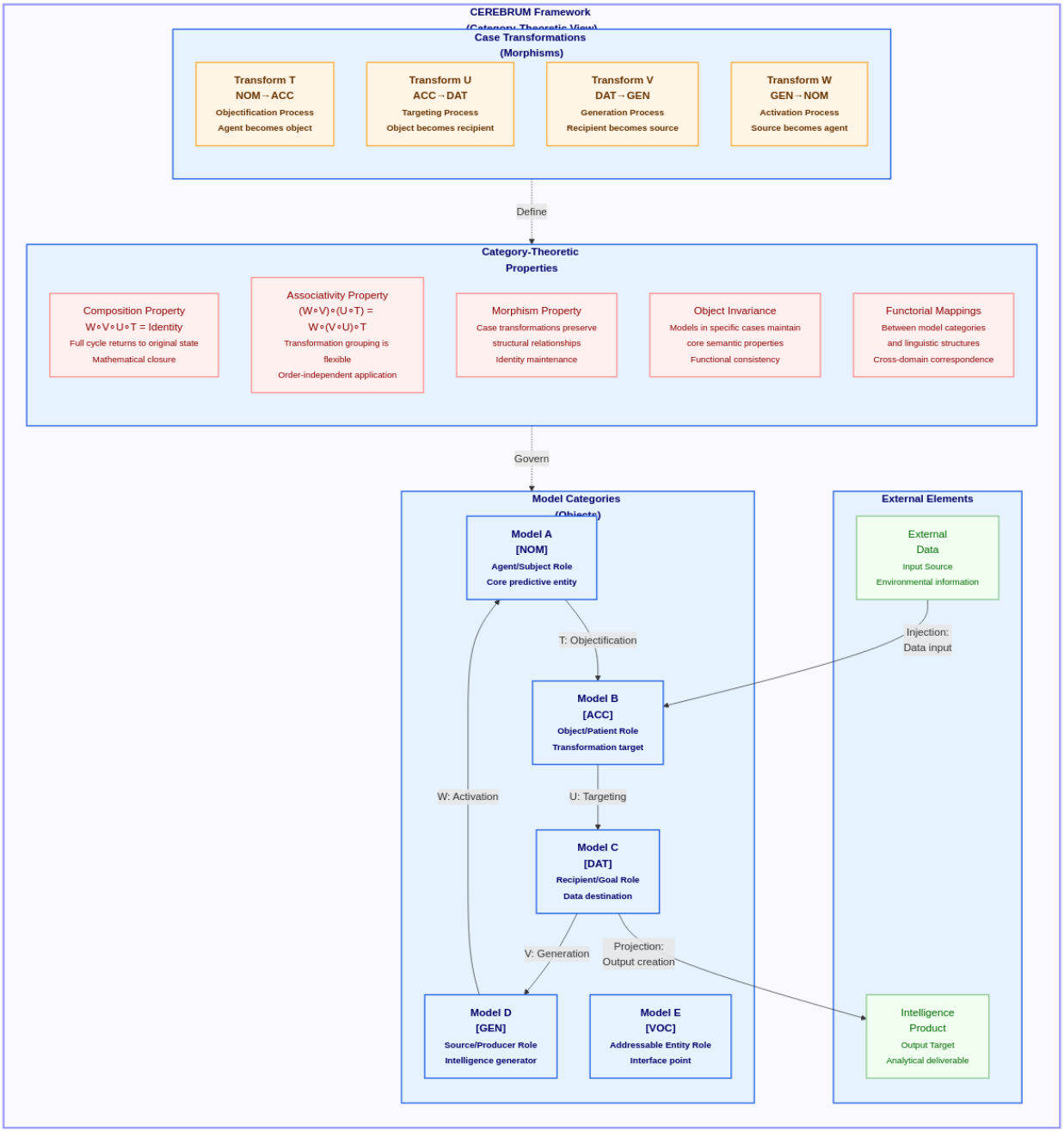


Figure 9: Figure 9. CEREBRUM Category Theory Framework. This diagram formalizes the CEREBRUM framework using category theory, providing a rigorous mathematical foundation for model transformations. The framework represents cognitive models as objects in a category, with different case assignments (Nominative, Accusative, Dative, Genitive, Vocative) defining their functional roles. Case transformations are represented as morphisms (T, U, V, W) between these objects, establishing principled pathways for models to change their functional roles while preserving their core identity. The diagram highlights critical category-theoretic properties: the composition property ensures that a full cycle of transformations returns a model to its original state (W∘V∘U∘T = Identity); the associativity property enables flexible grouping of transformations; morphism properties ensure that transformations preserve structural relationships; and object invariance maintains core semantic properties across case changes. External elements interact with the framework through injection (input) and projection (output) operations. This category-theoretic approach provides CEREBRUM with formal verification of properties like identity preservation and compositional consistency, enabling sound reasoning about model transitions in complex workflows.

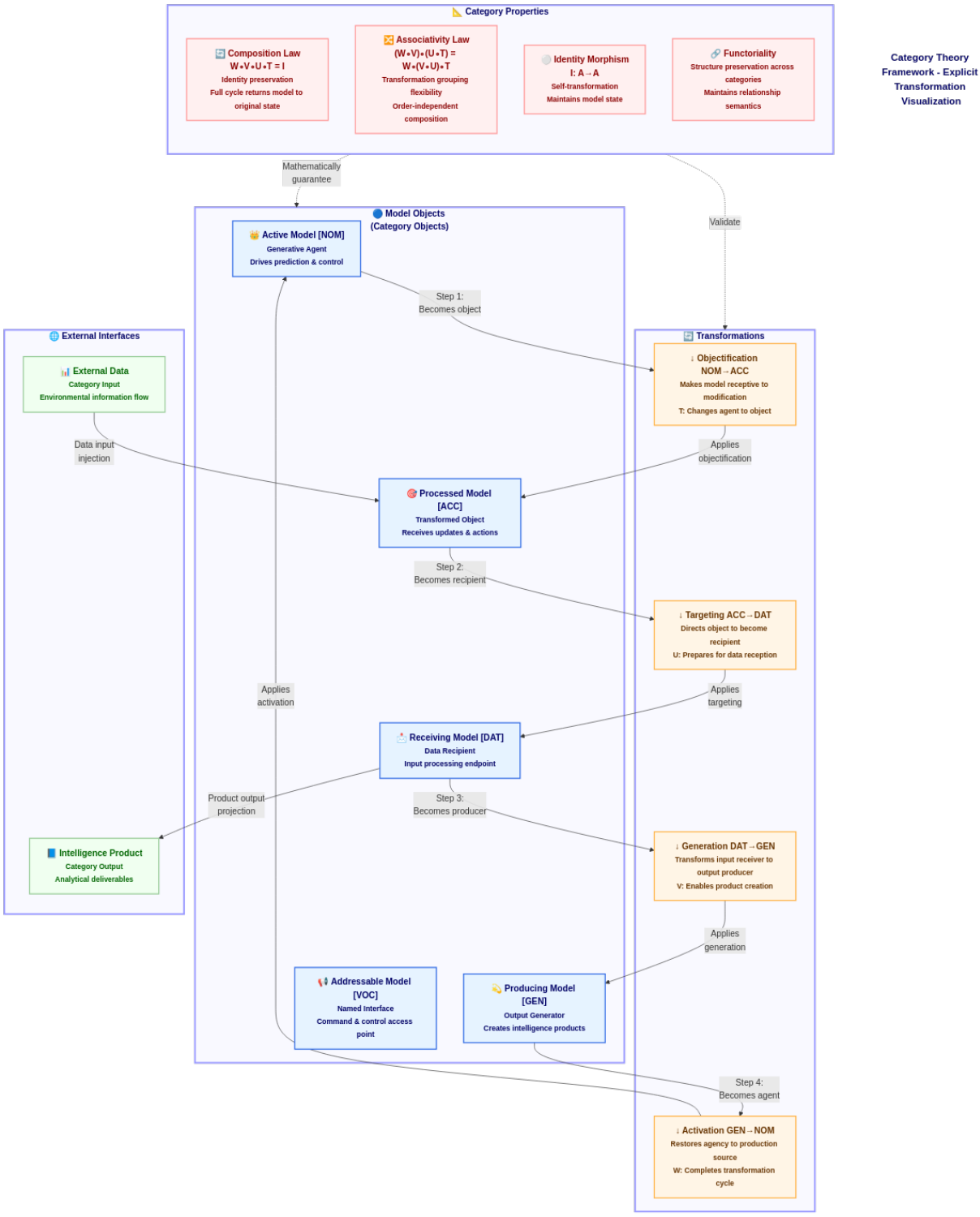


Figure 10: Figure 10. Category Theory Framework - Alternative Visualization. This diagram provides a complementary perspective to Figure 7, offering a more explicit representation of the transformations between case-bearing models in CEREBRUM. While Figure 7 presents the category-theoretic structure with a focus on properties and relationships, this visualization emphasizes the transformation process itself. Each case transformation (Objectification, Targeting, Generation, and Activation) is represented as a distinct node between model objects, clearly illustrating how models transition between different functional roles. The diagram highlights the cyclical nature of these transformations, with a model potentially moving from Nominative [NOM] through Accusative [ACC], Dative [DAT], and Genitive [GEN] cases before returning to its original state. Additionally, this representation explicitly shows the identity morphism (I: A→A) that maintains a model's state when no transformation is applied. External data enters the system through the Accusative case (as the object of processing),

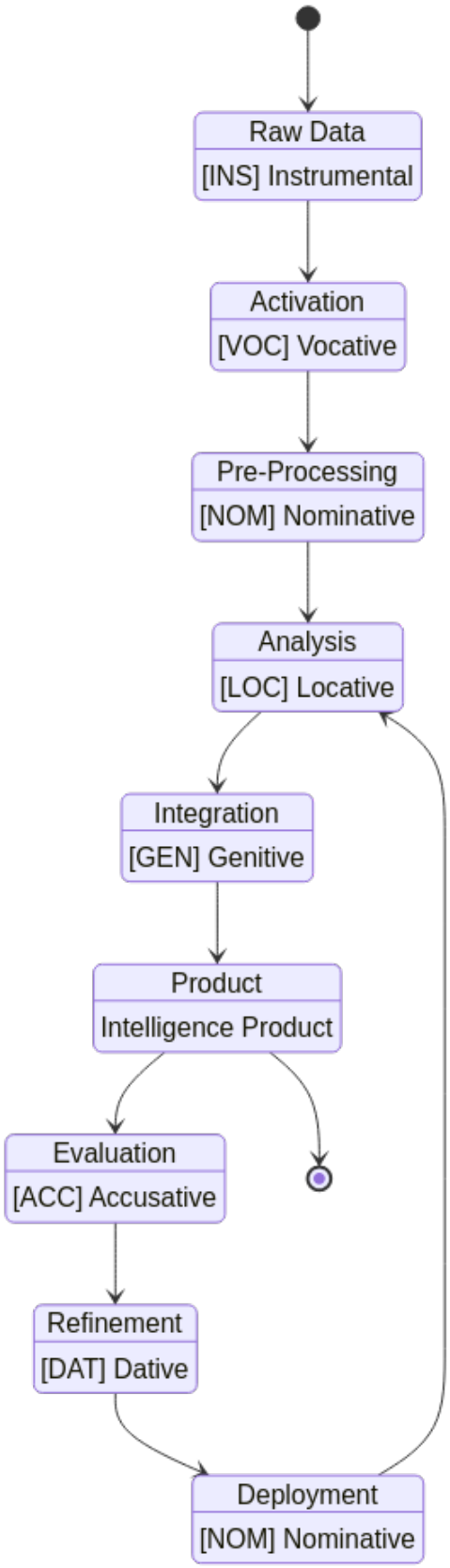


Figure 11: Figure 11. Intelligence Production Workflow State Diagram. This state diagram visualizes the transformation of data into intelligence products through a series of interconnected states. Beginning with raw data collection (Instrumental case [INS]), the process moves through system activation (Vocative case [VOC]), pre-processing (Nominative case [NOM]), analysis (Locative case [LOC]), and integration (Genitive case [GEN]), ultimately

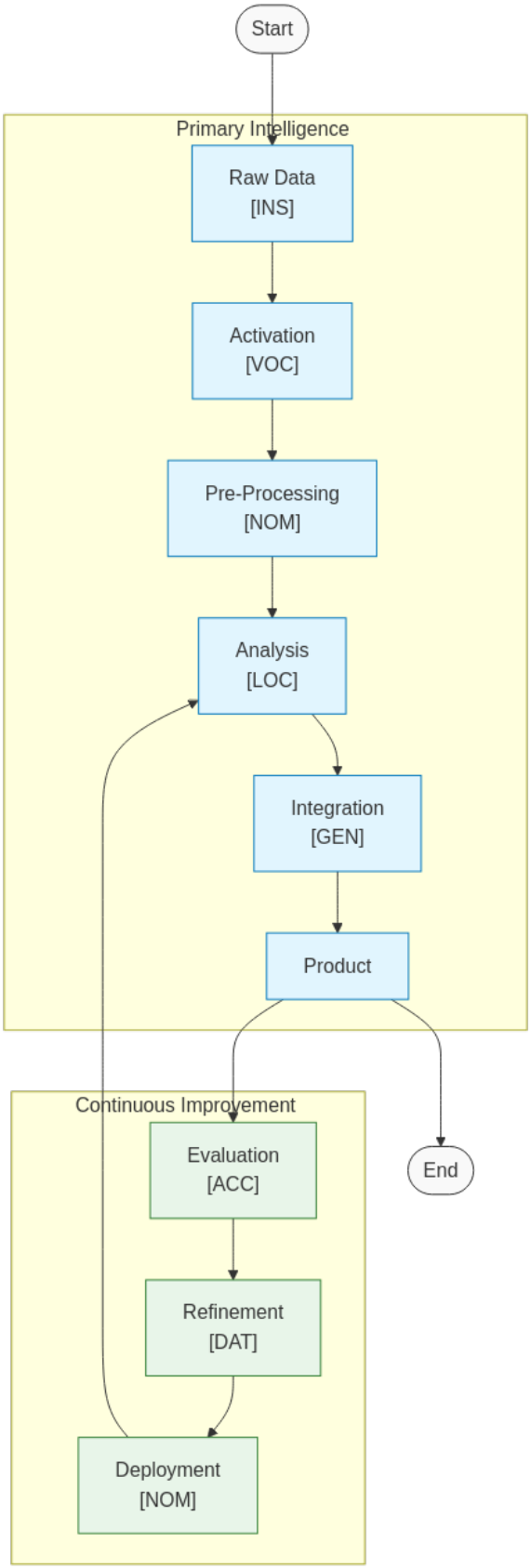


Figure 12: Figure 12. Alternative Visualization of Intelligence Production Workflow. This diagram presents a complementary perspective to Figure 11, organizing the intelligence workflow into two distinct components: the main Intelligence Workflow and a separate Feedback Loop. The main workflow proceeds linearly from raw data collection (using a model in Instrumental case [INS]) through system activation (Vocative case [VOC]), pre-processing (Nomi-

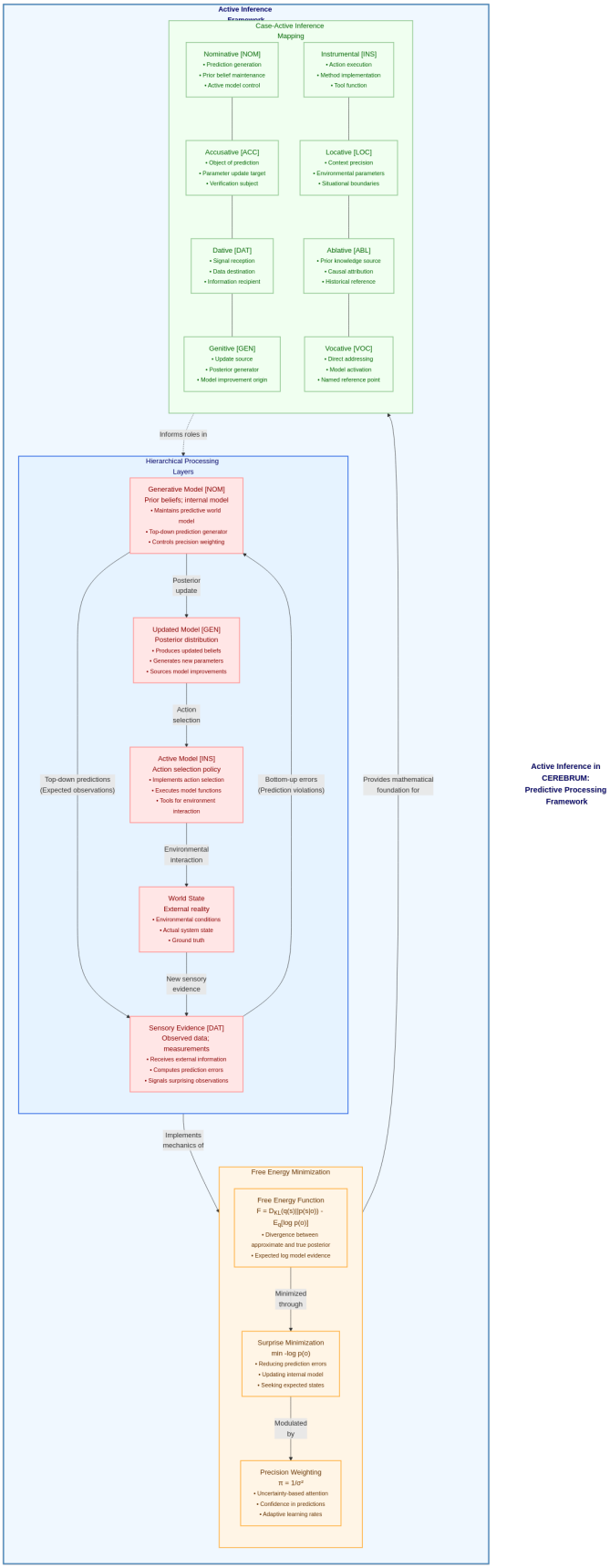


Figure 13: Figure 13. Active Inference Integration Framework. This diagram illustrates how CEREBRUM integrates with active inference principles, showing the mapping between case assignments and predictive processing mechanisms. The framework is organized into three interconnected components. The Processing Hierarchy demonstrates the core active inference cycle: a generative model in Nominative case [NOM] sends top-down predictions

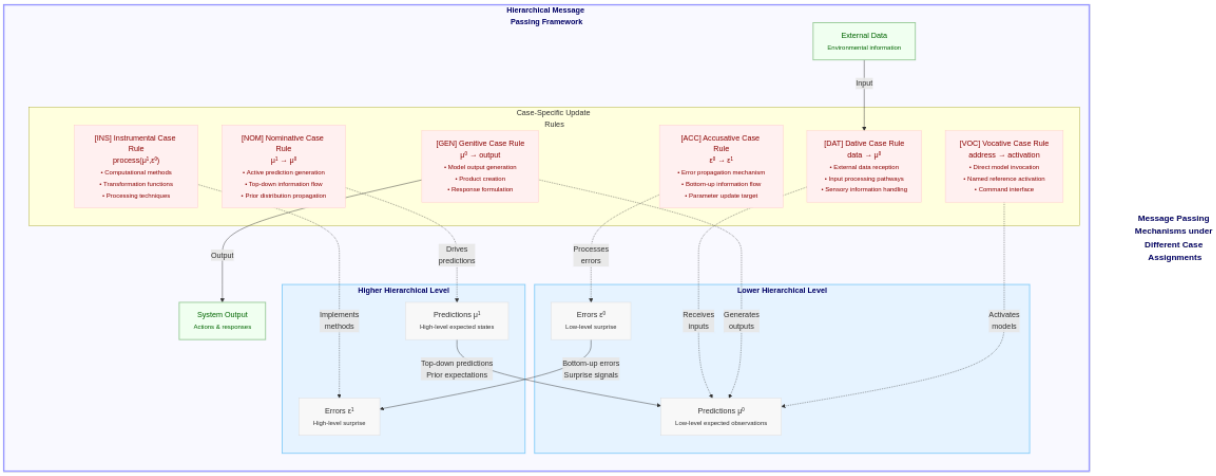


Figure 14: Figure 14. Case-Specific Message Passing in Active Inference. This diagram details the specific message passing mechanisms that implement case transformations within CEREBRUM's active inference framework. The central component shows the hierarchical bidirectional message passing that characterizes active inference, with top-down predictions (μ^1) flowing from higher to lower levels and bottom-up prediction errors (ε^0) propagating from lower to higher levels. The Case Rules section specifies how each case modulates these message flows: Nominative case [NOM] governs top-down prediction generation ($\mu^1 \rightarrow \mu^0$); Accusative case [ACC] handles bottom-up error propagation ($\varepsilon^0 \rightarrow \varepsilon^1$); Dative case [DAT] manages incoming data reception (data $\rightarrow \mu^0$); Genitive case [GEN] controls output generation ($\mu^0 \rightarrow$ output); Instrumental case [INS] implements processing functions that operate on both predictions and errors (process(μ^1,ε^0)); and Vocative case [VOC] manages direct activation through addressing (address $\rightarrow$ activation). These case-specific update rules create a functional specialization while maintaining the core active inference principles of prediction error minimization. By formalizing message passing in terms of case-specific operations, CEREBRUM provides a precise mathematical framework for implementing case transformations as precision-weighted Bayesian updates within hierarchical generative models. This approach connects linguistic case semantics directly to computational message-passing algorithms, creating a principled foundation for model interactions in complex cognitive systems.

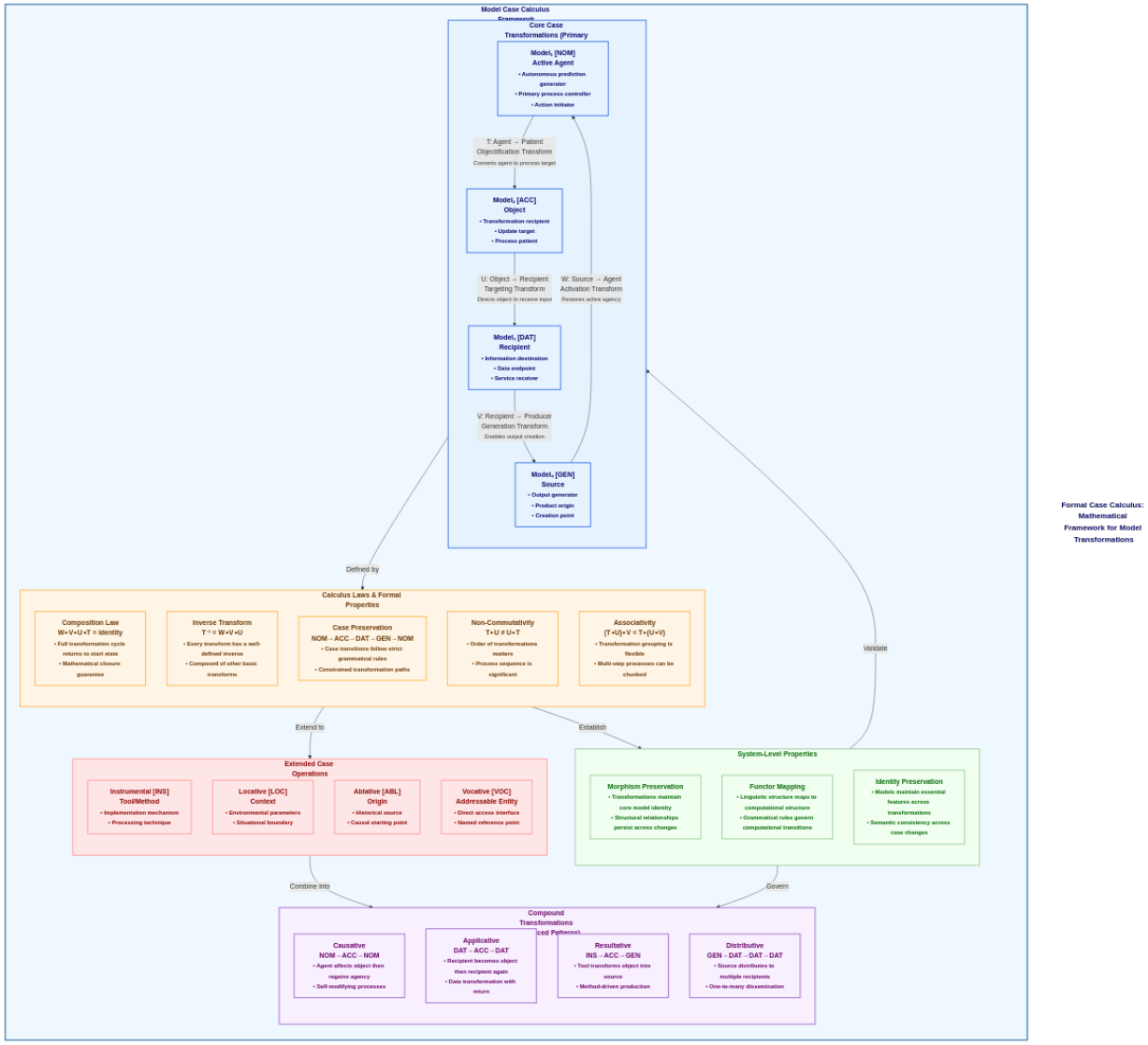


Figure 15: Figure 15. Model Case Calculus Framework. This diagram presents the formal mathematical calculus underlying CEREBRUM's case transformations, providing a rigorous foundation for modeling dynamic role changes in cognitive systems. The Core Case Transformations section shows the primary cycle between the four main cases: Nominative [NOM] (active agent), Accusative [ACC] (object), Dative [DAT] (recipient), and Genitive [GEN] (source). Each transformation (T, U, V, W) represents a specific morphism that changes a model's functional role while preserving its core identity. The Calculus Laws formalize these transformations mathematically: the Composition Law states that a complete cycle of transformations returns a model to its original state ($W \circ V \circ U \circ T$ = Identity); the Inverse Transform Law defines how to reverse any transformation; and the Case Preservation Law ensures consistent transition paths between cases. The System Properties section highlights mathematical characteristics of the framework: transformations are non-commutative (order matters), associative (grouping is flexible), and identity-preserving (full cycles maintain model identity). The Extended Operations section includes additional cases (Instrumental, Locative, Ablative, Vocative) that expand the framework's expressiveness. Finally, the Compound Transformations section demonstrates how basic transformations can be combined to create complex operations like causative, applicative, and resultative transformations. This formal calculus provides CEREBRUM with mathematical rigor, enabling consistent reasoning about model transitions, verifiable properties, and compositional guarantees in complex model ecosystems.

Domain	Contribution	Benefit to CEREBRUM	Theoretical Significance
Linguistic Case Systems	Provides systematic relationship frameworks, grammatical role templates, and morphosyntactic structures	Enables structured model interactions, formalizes functional transitions, and systematizes role assignments	Establishes formal semantics for model relationships and supports compositional theories of model interaction
Cognitive Systems Modeling	Offers entity representation methodologies, model formalization techniques, and information-processing structures	Facilitates flexible model instantiation, enables adaptive model morphology, and creates unified modeling paradigms	Advances cognitive model composition theory and formalizes functional transitions in cognitive architectures
Active Inference	Supplies predictive transformation mechanics, free energy principles, and precision-weighted learning methodologies	Creates self-optimizing workflows, enables principled uncertainty handling, and supports bidirectional message passing	Extends active inference to model ecosystems and provides mathematical foundations for case transformations
Intelligence Production	Contributes practical operational contexts, analytical workflows, and intelligence cycle formalisms	Enables real-world applications in case management, enhances operational coherence, and maintains analytical integrity	Bridges theoretical and applied intelligence domains and improves intelligence product quality

Related Work

CEREBRUM synthesizes concepts from several established research traditions. While this initial paper focuses on presenting the core framework, future work will elaborate on specific theoretical derivations and connections. CEREBRUM builds upon related approaches in the following areas:

Cognitive Architectures

CEREBRUM introduces a novel perspective on intelligent system design by using linguistic declension as inspiration for flexible model architectures with dynamic role assignment [1]. Unlike traditional cognitive architectures such as ACT-R and Soar where components have fixed functions [2], CEREBRUM enables cognitive models to adapt their roles via case transformations. This facilitates both flexibility and specialization, applying morphological transformations to generative models to create polymorphic components that maintain core identity while adapting interfaces, parameters, and dynamics to context [3]. This provides a principled foundation for coordinating complex model ecosystems, as detailed in Supplement 3.

Category-Theoretic Cognition

The formalization of CEREBRUM using category theory provides a basis for compositional reasoning about cognitive systems [4]. By representing case transformations as functors between categories of models, the framework enables formal verification of properties like identity preservation across functional transitions [5]. This supports reasoning about model composition, transformation sequencing, and structural relationships, aligning with the compositional nature of cognition [6]. These category-theoretic foundations are mathematically formalized in Supplement 9.

Active Inference Applications

CEREBRUM extends active inference from individual processes to entire model ecosystems [7]. By treating case transformations as precision-weighted processes minimizing free energy across the system [8], the framework implements principled coordination by explicitly representing functional relationships through case assignments and formalizing interfaces with precision-weighting [9]. This creates intelligent workflows where models cooperate to minimize system-wide free energy, with the formal mathematical definitions provided in Supplement 11.

Linguistic Computing

CEREBRUM exemplifies a linguistic computing approach, applying declensional semantics to model management [10]. By treating models as entities assuming different morphological forms based on functional roles—mirroring noun declension in natural languages [11]—the framework implements computable declension paradigms allowing for more expressive and flexible representations of model relationships than traditional paradigms [12]. This bridges natural and artificial intelligence systems through shared linguistic principles, as explored in Supplement 2.

See Supplement 2 for a discussion on novel linguistic cases, Supplement 3 for practical implementation examples, and Supplement 7 for analysis of computational complexity across different case assignments.

Conclusion

CEREBRUM advances model management by applying linguistic case principles to cognitive systems, establishing three primary contributions: (1) a formal framework for representing functional model roles through case-based transformations, (2) a mathematical basis for model transitions rooted in category theory and variational principles, and (3) practical implementation patterns for intelligence production workflows. This integration of linguistic theory, category mathematics, active inference, and intelligence production creates a systematic approach to complex model ecosystems. By treating models as case-bearing entities, CEREBRUM enables precise transformations between model states while providing structured representations of model relationships that align with human cognitive patterns and operational intelligence workflows.

The formal integration of variational free energy principles with case transformations, detailed in Supplement 11, establishes CEREBRUM as a mathematically rigorous framework for active inference implementations. The precision-weighted case selection mechanisms, Markov blanket formulations, and hierarchical message passing structures provide computationally tractable algorithms for optimizing model interactions. These technical formalizations bridge theoretical linguistics and practical cognitive modeling while maintaining mathematical coherence through category-theoretic validation as presented in Supplement 9.

The CEREBRUM framework represents a significant advancement in model relationship conceptualization, moving from ad hoc integration approaches toward principled foundations for persistent, composable, linguistic intelligences. By formalizing the grammatical structure of model interactions, CEREBRUM enhances current capabilities and opens new avenues for modeling emergent behaviors in ecosystems of shared intelligence. As computational systems continue to grow in complexity, frameworks like CEREBRUM that provide structured approaches to model management will become increasingly essential for maintaining conceptual coherence and operational effectiveness.

Future research will explore three specific directions: (1) extending the case framework to include additional linguistic structures such as aspect, mood, and voice; (2) implementing computational libraries that enable automatic case transformations in high-performance computing environments; and (3) applying the framework to multi-agent systems where case transformations facilitate coordinated collective intelligence. These developments will further establish CEREBRUM as both a theoretical framework and practical methodology for next-generation cognitive systems.

The CEREBRUM approach demonstrates that linguistic principles—specifically those governing case relationships—provide powerful organizing structures for complex model ecosystems. By bridging human language and machine intelligence through shared grammatical patterns, CEREBRUM creates a foundation for more intuitive, expressive, and adaptable intelligent systems that can assume different functional roles while maintaining coherent identities across transformations.

Supplement 1: Mathematical Formalization

This supplement contains all mathematical formalizations referenced throughout the paper, organized by equation number.

1.1 Variational Free Energy Framework

CEREBRUM builds on the foundational concept of variational free energy in active inference, establishing a principled approach to model transformations. The variational free energy F for a model m with internal states s and observations o is:

$$F = D_{KL}[q(s|T(m))||p(s|m)] - \mathbb{E}_p[\log p(o|s, T(m))]$$

This formulation, where $T(m)$ represents a case transformation applied to model m, captures two essential quantities: (1) the KL divergence between the recognition density and prior, measuring complexity; and (2) the expected log-likelihood, measuring accuracy. Together, they form the evidence lower bound that models seek to minimize.

1.2 Markov Blanket Formulation

A key insight in CEREBRUM is the interpretation of case transformations as operations on Markov blankets. The Markov blanket of a model M contains all variables that shield it from the rest of the network, comprising parents, children, and co-parents. We define:

$$\text{Case}(M) \subseteq \text{MB}(M)$$

This indicates that a case assignment acts on a subset of the Markov blanket, modifying how information flows between the model and its environment.

1.3 Precision-Weighted Case Selection

The probability of a model assuming a particular case c is determined by its ability to minimize free energy. The softmax function provides a natural mechanism for case selection:

$$\beta(c, m) = \frac{\exp(-F(c, m))}{\sum_i \exp(-F(c_i, m))}$$

Where $\beta(c, m)$ represents the probability of model m adopting case c. This allows for dynamic case assignment based on contextual factors and observations.

1.4 Dynamical Implementation

The dynamics of case transformations can be implemented through gradient flows on free energy:

$$\frac{\partial m}{\partial t} = -\kappa_c \cdot \frac{\partial F}{\partial m}$$

Here, κ_c is a case-specific learning rate that determines how quickly the model adapts when in a particular case. This provides a continuous-time formulation of case-based learning.

1.5 Expected Free Energy Minimization

For planning and policy selection, CEREBRUM extends to expected free energy minimization over sequences of case transformations:

$$\mathbb{E}[\Delta F] = \sum_{s,a} T(s'|s,a)\pi[a|s](F(s,c) - F(s',c'))$$

Where $T(s'|s,a)$ is the transition probability from state s to s' given action a, and $\pi[a|s]$ is the policy. This allows for optimal sequencing of case transformations to achieve goals.

1.6 Bayesian Model Comparison Between Cases

To evaluate competing case assignments, we employ Bayesian model comparison using the Bayes Factor:

$$BF = \frac{p(o|m,c_1)}{p(o|m,c_2)}$$

This quantifies the relative evidence for model m being in case c_1 versus case c_2 given observations o.

1.7 Case-Specific Free Energy

The case-specific free energy explicitly includes the case c in the formulation:

$$F = D_{KL}[q(s|c,m)||p(s|m)] - \mathbb{E}_{q(s|c,m)}[\log p(o|s,c,m)]$$

This allows different cases to maintain distinct recognition densities and likelihood functions while operating on the same underlying model.

1.8 Core Linguistic Case Equations

CEREBRUM defines mathematical operations for each core linguistic case, establishing precise transformations:

$$\text{Nominative [NOM]} : \mu^0 = \mu^0 + \kappa_{NOM} \cdot (\mu^1 - \mu^0)$$

(Lower-level prediction μ^0 updated by top-down prediction μ^1, weighted by κ_{NOM})

$$\text{Accusative [ACC]} : \varepsilon^1 = \varepsilon^1 + \kappa_{ACC} \cdot (\varepsilon^0 - \varepsilon^1)$$

(Higher-level error ε^1 updated by bottom-up error ε^0, weighted by κ_{ACC})

$$\text{Dative [DAT]} : \mu^0 = \mu^0 + \kappa_{DAT} \cdot (data - \mu^0)$$

(Lower-level prediction μ^0 updated directly by incoming 'data', weighted by κ_{DAT})

$$\text{Genitive [GEN]} : output = \mu^0 + \kappa_{GEN} \cdot \eta$$

(Output generated based on lower-level prediction μ^0, weighted by κ_{GEN} and noise η)

$$\text{Instrumental [INS]} : process = f(\mu^1, \varepsilon^0) \cdot \kappa_{INS}$$

(Process defined as a function of top-down prediction and bottom-up error, scaled by κ_{INS})

$$\text{Vocative [VOC]} : activation = \sigma(\kappa_{VOC} \cdot sim(id, address))$$

(Activation determined by similarity between model's identity and the addressing signal, weighted by κ_{VOC})

1.9 Precision-Weighted Mixture of Cases

Models can simultaneously exist in multiple cases with varying probabilities. The case probability is precision-weighted:

$$\beta(c, m) = \frac{\exp(-\gamma \cdot F(c, m))}{\sum_i \exp(-\gamma \cdot F(c_i, m))}$$

Where γ represents the precision or confidence in case assignments. This enables nuanced, multimodal model behavior.

1.10 Composite Free Energy

The effective free energy for a model in a mixture of cases is:

$$F_\beta(m) = \sum_c \beta(c, m) \cdot F(c, m) \cdot R(c)$$

Where $R(c)$ represents the relevance or priority of case c. This formulation allows for weighted importance across different case assignments.

1.11 Conjunction of Models

For composite systems combining multiple models, the Conjunctive case [CNJ] has a specialized free energy:

$$F_{CNJ} = D_{KL}[q(s|CNJ, m)||p(s|m)] - \mathbb{E}_{q(s|CNJ,m)}[\log p(o|s, \{m_i\})]$$

This formulation accounts for emergent properties arising from model interactions, where $\{m_i\}$ represents the set of constituent models.

1.12 Conjunctive Mean Distribution

The mean prediction in the Conjunctive case is:

$$\mu^{CNJ} = \sum_i w_i \cdot \mu_i + \kappa_{CNJ} \cdot (\prod_i \mu_i - \sum_i w_i \cdot \mu_i)$$

This combines weighted averages with multiplicative interactions, controlled by κ_{CNJ}, representing nonlinear emergent effects.

1.13 Recursive Case Formulation

The Recursive case [REC] allows for self-reference, with probability:

$$\beta(REC, m) = \frac{\exp(-\gamma \cdot F(REC, m))}{\sum_i \exp(-\gamma \cdot F(c_i, m)) + \exp(-\gamma \cdot F(REC, m))}$$

This creates a distinct pathway for models to operate on themselves, enabling self-modification capabilities.

1.14 Glossary of Variables

- a: Action (in MDP context, often selecting a case transition)
- α: Learning rate (in Neural Process Models context)
- BF: Bayes Factor (for comparing model evidence between cases)
- c, c_i, c', c_1, c_2: Linguistic case assignment (e.g., NOM, ACC, specific case instances)
- Case(M): Case assignment of model M
- **Case Transformation**: An operation that changes the functional role (case) of a model within the system
- **CEREBRUM**: Case-Enabled Reasoning Engine with Bayesian Representations for Unified Modeling
- D_{KL}: Kullback-Leibler divergence
- data: Input data (in Dative case message passing; Eq 10)
- **Declinability**: The capacity of a generative model within CEREBRUM to assume different morphological and functional roles (cases) through transformations
- $E_p[\cdot]$: Expectation with respect to distribution p (Information Geometry)
- $\mathbb{E}[\cdot]$: Expectation operator
- F: Variational Free Energy
- $F_\beta(m)$: Resource-weighted free energy for model m
- F_{CNJ}: Free energy for the speculative Conjunctive case
- $f(...)$: Function (used generally; e.g., in Instrumental message passing; Eq 12)
- g_{ij}: Fisher information metric tensor component (Information Geometry)
- i, j: Indices for summation or tensor components
- $L(M)$: Lyapunov function for model M (Dynamical Systems section)
- m, M: Cognitive model
- $\{m_i\}$: Assembly or set of connected models
- MB(M): Markov blanket of model M
- **Morphological Marker (Computational Analogue)**: Specific computational properties (e.g., active interfaces; parameter access patterns; update dynamics) that signal a model's current case assignment within CEREBRUM
- n: Model parameter count (Complexity section)
- $O(...)$: Big O notation for computational complexity
- o: Observations or sensory data
- output: Output generated by a model (in Genitive case; Eq 11)
- $p(s|...)$: Prior distribution over internal states s
- $p(o|...)$: Likelihood distribution of observations o
- $p(x|theta)$: Probability distribution of data x given parameters $theta$ (Information Geometry)
- process: Result of a process executed by a model (in Instrumental case; Eq 12)
- $q(s|...)$: Approximate posterior distribution over internal states s
- $R(c)$: Computational resources allocated to case c
- REC: Speculative Recursive case assignment

- s: Internal states of a model
- s': Next state (in MDP context; target case assignment)
- t: Time variable (in gradient descent context; Eq 4)
- T: Transformation function (e.g., $T(m)$ is a transformed model in Eq 1; also MDP transition function)
- $T(s'|s,a)$: State transition function in MDP (probability of transitioning to state s' from state s given action a)
- w_i: Model-specific weighting factors (in Conjunctive case; Eq 16)
- ΔF: Change in Free Energy
- Δw_{ij}: Change in synaptic weight between neuron i and j (Neural Process Models section)
- $\beta(c,m)$: Precision weight (allocation) assigned to model m in case c
- γ: Inverse temperature parameter (controlling precision allocation sharpness)
- ϵ_i: Error signal of neuron i (Neural Process Models section)
- $\varepsilon^0, \varepsilon^1$: Error signals used in message passing (representing prediction errors at adjacent hierarchical levels; Eq 9, 12)
- η: Noise term (Eq 11)
- κ_c: Case-specific learning rate or precision weight (modulating message updates; Eqs 4, 8-12)
- μ^0, μ^1: Mean values used in message passing (representing predictions or beliefs at adjacent hierarchical levels)
- μ^{CNJ}: Mean value resulting from Conjunctive case message passing
- $\pi(a|s)$: Policy in MDP (probability of taking action a in state s)
- $\sigma'(a_j)$: Derivative of activation function of neuron j (Neural Process Models section)
- $theta, theta_i, theta_j$: Model parameters

Supplement 2: Novel Linguistic Cases

This supplement explores new linguistic cases that emerge from the CEREBRUM framework, examining how the cognitive declension paradigm generates novel case functions not found in traditional linguistics. These cases demonstrate the framework's ability to formalize new types of model relationships.

2.1 Discovering and Creating New Linguistic Cases Through CEREBRUM

The CEREBRUM framework not only operationalizes traditional linguistic cases but potentially enables the discovery of entirely new case archetypes through its systematic approach to model interactions. As cognitive models interact in increasingly complex ecosystems, emergent functional roles may arise that transcend the classical case system derived from human languages.

Table A0: Overview of Novel Linguistic Cases in CEREBRUM

Case Name	Symbol	Primary Function	Key Innovation	Paradigmatic Application
Conjunctive	[CNJ]	Synthesizes multiple predictive streams into coherent joint predictions	Integration of diverse model outputs through multiplicative fusion	Multi-modal prediction systems; ensemble learning architectures; consensus-building mechanisms
Recursive	[REC]	Applies transformations to itself, enabling computational reflection	Self-reference mechanisms allowing models to modify their own parameters	Self-improving systems; meta-learning; neural architecture search; introspective AI
Metaphorical	[MET]	Maps structures and relationships between disparate domains	Cross-domain structural alignment preserving relational invariants	Transfer learning; analogical reasoning; creative problem-solving; interdisciplinary modeling
Explicative	[EXP]	Translates internal model representations into human-interpretable formats	Bidirectional mapping between model internals and explanatory abstractions	Explainable AI; model debugging; regulatory compliance; transparent decision systems
Diagnostic	[DIA]	Identifies, localizes, and characterizes model pathologies and anomalies	Systematic comparison between expected and actual model behaviors	AI safety; model robustness testing; adversarial defense; quality assurance

Case Name	Symbol	Primary Function	Key Innovation	Paradigmatic Application
Orchestrative	[ORC]	Coordinates model ensembles and allocates computational resources	Context-sensitive resource allocation based on task requirements	Edge computing; distributed AI; multi-agent systems; efficient ML deployment
Generative	[GEN]	Creates novel yet coherent instances within a learned distribution	Controlled sampling from latent spaces with directed constraints	Content creation; hypothesis generation; synthetic data augmentation; design ideation

2.2 Emergence of Novel Case Functions

Traditional linguistic case systems evolved to serve human communication needs in physical and social environments. However, computational cognitive ecosystems face novel challenges and opportunities that may drive the emergence of new functional roles. The mathematical formalism of CEREBRUM provides a scaffold for identifying these emergent case functions through:

1. **Pattern detection in model interaction graphs**: Recurring patterns of information flow that don't fit established cases
2. **Free energy anomalies**: Unusual optimization patterns indicating novel functional configurations
3. **Precision allocation clusters**: Statistical clustering of precision weightings revealing new functional categories
4. **Transition probability densities**: Dense regions in case transition probability spaces suggesting stable new cases

2.3 The Conjunctive Case [CNJ]

The conjunctive case represents a model's role in synthesizing multiple predictive streams into coherent joint predictions that couldn't be achieved through simple composition of existing cases. Unlike traditional aggregation methods, the conjunctive case implements sophisticated fusion mechanisms that preserve the correlational structure across predictive streams.

The mathematical formalism for a model in conjunctive case would extend the standard free energy equation as shown in Equation 15 (see Supplement 1), representing the assembly of connected models participating in the joint prediction. The key innovation is that the likelihood term explicitly depends on multiple models' predictions rather than a single model's output, enabling integration of diverse predictive streams.

In the message-passing formulation, the conjunctive case would introduce unique update rules as described in Equation 16 (see Supplement 1), with weighting factors for individual model predictions, as well as a multiplicative integration of predictions that captures interdependencies beyond simple weighted averaging. This formulation enables rich joint inference across model collectives.

2.3.1 Unique Properties of the Conjunctive Case

The conjunctive case is distinguished by its ability to maintain coherence across multiple streams of information while optimizing for global consistency. It acts as an integration hub

in model assemblies, establishing higher-order constraints that guide the collective behavior of connected models.

Table CNJ: Comprehensive Details of the Conjunctive Case [CNJ]

Aspect	Description	Mathematical Formulation	Examples & Applications
Core Function	Synthesizes multiple model outputs into coherent joint predictions that preserve correlational structure	F[CNJ] = E[∑ wi log p(y|xi, θi)] - D[q(x|θ) || ∏ p(xi)]	Multi-sensor fusion; multi-expert systems; cross-modal alignment
Information Flow	Many-to-one convergent flow with bidirectional feedback	{xi → M[CNJ] → y} with {M[CNJ] ↔ Mi} feedback loops	Sensory integration; committee machines; ensemble methods
Error Handling	Resolves inconsistencies across predictive streams while minimizing global prediction error	ε[CNJ] = y - f(∑ wi·xi) with consistency penalty λ·∑ D[xi || xj]	Anomaly detection; conflict resolution; coherence optimization
Learning Dynamics	Updates based on joint prediction accuracy and inter-model consistency	Δθ[CNJ] ∝ -∇θ(prediction_error + λ·consistency_error)	Meta-ensemble training; cooperative multi-agent learning
Precision Allocation	Higher precision to consistent model outputs; dynamically weighted integration	π[CNJ] = ∑ αi·πi where αi ∝ exp(-βi·εi²)	Robust sensor fusion; adaptive information integration
Computational Complexity	O(n²) scaling with number of models due to pairwise consistency checks	Optimization requires evaluating n·(n-1)/2 pairwise constraints	Hierarchical clustering to reduce computation; sparse interaction approximations
Representation Requirements	Shared latent space or translation mechanisms between models	Requires alignment functions f_ij: Xi → Xj for each model pair	Cross-modal embeddings; shared semantic spaces

Aspect	Description	Mathematical Formulation	Examples & Applications
Biological Analogues	Multi-sensory integration areas in brain (e.g., superior colliculus)	Bayesian integration with cross-modal priors	Visual-auditory integration; sensorimotor coordination
Implementation Approaches	Product-of-experts; mixture density networks; attention-based fusion	p(y|x1,...,xn) ∝ ∏ p(y|xi)^wi / Z	Transformer attention mechanisms; graph neural networks; hypernetworks
Failure Modes	Susceptible to reinforcing consistent but incorrect predictions	Can amplify correlated errors across models	Echo chambers; confirmation bias; groupthink

2.4 The Recursive Case [REC]

The recursive case enables a model to apply its transformations to itself, creating a form of computational reflection not captured by traditional cases. This case introduces self-reference into the CEREBRUM framework, allowing models to introspect and modify their own parameters through directed self-transformations.

In the recursive case, a model assumes both agent and object roles simultaneously, creating feedback loops that enable complex self-modification behaviors. This case would be particularly relevant for metalearning systems and artificial neural networks that modify their own architectures.

The recursive case would introduce unique precision dynamics as formalized in Equation 17 (see Supplement 1). The key innovation is that the model appears on both sides of the transformation, creating a form of self-reference that traditional case systems don't accommodate. This enables models to introspect and modify their own parameters through self-directed transformations.

2.4.1 Unique Properties of the Recursive Case

The recursive case establishes a formal framework for self-improvement and meta-cognition in computational systems. It provides mechanisms for models to observe their own operations, evaluate their performance, and systematically modify themselves to improve outcomes.

Table REC: Comprehensive Details of the Recursive Case [REC]

Aspect	Description	Mathematical Formulation	Examples & Applications
Core Function	Enables models to modify themselves through self-directed transformations	M' = M(M, θ) where M is both operator and operand	Self-modifying code; neural architecture search; autoencoder networks

Aspect	Description	Mathematical Formulation	Examples & Applications
Information Flow	Circular reflexive flow creating feedback loops within a single model entity	M → M with self-reference implemented through level indicators	Self-attention mechanisms; recursive neural networks; introspection systems
Error Handling	Multi-level error tracking that differentiates between object-level and meta-level errors	ε[REC] = ε[object] + λ·ε[meta] where ε[meta] evaluates self-modification quality	Stack traces; recursive debugging; hierarchical error detection
Learning Dynamics	Interleaved learning at multiple levels of recursion; gradient flow through self-reference paths	θ[level i+1] updated based on performance of θ[level i] modifications	Meta-learning; learning-to-learn; hyperparameter optimization
Precision Allocation	Precision hierarchies with meta-level precision controlling object-level precision distributions	π[level i+1] governs the allocation of π[level i]	Attention over attention; multi-level uncertainty estimation
Computational Complexity	Potentially unbounded without proper termination conditions; practical implementations require recursion limits	Complexity grows with recursion depth d as O(f^d) for base complexity f	Stack overflow prevention; recursion depth monitoring; fixed-point detection
Representation Requirements	Requires homomorphic self-representation and differentiation between levels of recursion	Model must maintain (M, M(M), M(M(M)), ...) distinctions	Type systems with self-types; reflection in programming languages

Aspect	Description	Mathematical Formulation	Examples & Applications
Biological Analogues	Metacognition in human reasoning; prefrontal cortex monitoring other brain regions	Hierarchical predictive coding with top levels modeling lower levels	Executive function; self-regulation; introspective awareness
Implementation Approaches	Higher-order derivatives; neural Turing machines; recursive neural architectures	Implemented via operator fixed points or higher-order derivatives	Gödel numbering systems; lambda calculus implementations; hypernetworks
Failure Modes	Infinite recursion; meta-stable self-modification cycles; self-reinforcing errors	Can develop optimization pathologies or recursive traps	Overthinking; analysis paralysis; infinite regress

2.5 The Metaphorical Case [MET]

The metaphorical case enables a model to map structures and relationships from one domain to another, creating computational analogies that transfer knowledge across conceptual spaces. This case establishes formal mechanisms for analogical reasoning and cross-domain knowledge transfer within the CEREBRUM framework.

In the metaphorical case, a model acts as a transformation bridge between disparate domains, establishing systematic mappings between conceptual structures. This case would be particularly valuable for transfer learning systems and creative problem-solving algorithms that need to apply learned patterns in novel contexts.

The metaphorical case would introduce unique cross-domain mapping functions as formalized in Equation 18 (see Supplement 1). The key innovation is the structured alignment of latent representations across domains, enabling principled knowledge transfer that preserves relational invariants while adapting to target domain constraints.

2.5.1 Unique Properties of the Metaphorical Case

The metaphorical case provides a formal framework for analogical reasoning and structural mapping across domains. It establishes mechanisms for identifying and preserving deep structural similarities while adapting surface features to target domain constraints.

Table MET: Comprehensive Details of the Metaphorical Case [MET]

Aspect	Description	Mathematical Formulation	Examples & Applications
Core Function	Maps structural relationships from source to target domains while preserving key invariants	M[MET]: X → Y such that R_X(x1, x2) ≈ R_Y(M(x1), M(x2)) for relations R	Analogical reasoning; transfer learning; cross-domain mapping
Information Flow	Bidirectional mapping between source and target domains with selective transfer of structure	X ⟷ M[MET] ⟷ Y with structure-preserving constraints	Domain adaptation; knowledge distillation; scientific modeling
Error Handling	Balances structure preservation error against target domain constraints	ε[MET] = w1·structural_error + w2·target_domain_error	Metaphor coherence checking; analogical validation; transfer relevance assessment
Learning Dynamics	Updates based on successful transfer outcomes and structural preservation quality	Δθ[MET] ∝ -∇θ(structure_preservation_error + λ·transfer_outcome_error)	Few-shot learning; zero-shot transfer; analogical bootstrapping
Precision Allocation	Highest precision on relational invariants; lower precision on surface features	π[relations] » π[attributes] in computing mapping costs	Structural correspondence finding; invariant detection
Computational Complexity	NP-hard in general case (subgraph isomorphism); practical approximations via embedding similarity	Typically $O(n^3)$ for structure mapping with heuristic constraints	Graph matching algorithms; structure mapping engines

Aspect	Description	Mathematical Formulation	Examples & Applications
Representation Requirements	Requires relational representations and structure-sensitive distance metrics	Domain models must expose structural interfaces for mapping	Knowledge graphs; relation networks; structure-sensitive embeddings
Biological Analogues	Human analogical reasoning; conceptual blending; cognitive metaphors	Hofstadter's fluid concepts; Lakoff's conceptual metaphors	Spatial reasoning applied to time; embodied cognition; conceptual metaphors
Implementation Approaches	Structure mapping engines; relation networks; analogical neural networks	Implemented via graph matching or embedding alignment with relational constraints	Structure mapping theory implementations; relational reinforcement learning
Failure Modes	False analogies; surface-level mapping; over-extension	Can extract spurious patterns or transfer irrelevant structures	Inappropriate transfer; false equivalences; misleading analogies

2.6 Connections to Human Cognition and Communication

The metaphorical case has rich connections to multiple domains of human cognition and communication. In affective neuroscience, it models how emotional experiences are mapped onto conceptual frameworks, explaining how we understand emotions through bodily metaphors (e.g., "heavy heart," "burning anger"). In first and second-person neuroscience, metaphorical mappings enable perspective-taking and empathy through systematic projection of one's own experiential models onto others. Educational contexts leverage metaphorical case operations when complex concepts are taught through familiar analogies, making abstract ideas concrete through structured mappings. The way people converse about generative models often employs metaphorical language—describing models as "thinking," "imagining," or "dreaming"—which represents a natural metaphorical mapping between human cognitive processes and computational operations. Learning itself fundamentally involves metaphorical operations when knowledge from one domain scaffolds understanding in another. Perhaps most profoundly, the metaphorical case provides a computational framework for understanding how symbols and archetypes function in human cognition—as cross-domain mappings that compress complex experiential patterns into transferable, culturally-shared representations that retain their structural integrity across diverse contexts while adapting to individual interpretive frameworks.

2.7 Implications of Novel Cases for Computational Cognition

The discovery of novel cases through CEREBRUM could have profound implications for computational cognitive science:

1. **Expanded representational capacity**: New cases enable representation of functional relationships beyond traditional linguistic frameworks
2. **Enhanced model compositionality**: Novel cases might enable more efficient composition of complex model assemblies
3. **Computational reflection**: Cases like the recursive case enable systematic implementation of self-modifying systems
4. **Cross-domain integration**: New cases like the metaphorical case might bridge domains that are difficult to connect with traditional case systems

These speculative extensions of CEREBRUM highlight its potential not just as an implementation of linguistic ideas in computational contexts, but as a framework that could expand our understanding of functional roles beyond traditional linguistic categories. The mathematical rigor of CEREBRUM provides a foundation for systematically exploring this expanded space of possible case functions, potentially leading to entirely new paradigms for understanding complex model interactions in cognitive systems.

2.8 Synergistic Combinations of Novel Cases

The novel cases introduced in CEREBRUM can be combined in powerful synergistic ways to address complex cognitive challenges. For example, a recursive-metaphorical [REC-MET] combination would enable systems that can reflect on their own analogical mapping processes, potentially creating higher-order metaphors and meta-analogies. Similarly, a conjunctive-metaphorical [CNJ-MET] combination could integrate multiple analogical mappings into cohesive knowledge transfers across complex domain clusters, enabling richer multi-domain knowledge synthesis.

The introduction of additional novel cases creates even more powerful combinatorial possibilities. An explicative-diagnostic [EXP-DIA] combination would enable systems that not only identify model pathologies but can explain them in human-interpretable terms. An orchestrative-generative [ORC-GEN] combination could coordinate distributed creative processes across multiple specialized generative models, enabling scalable and diversified content creation.

Table A2: Extended Properties of Novel Cases in CEREBRUM

Property	Conjunctive Case [CNJ]	Recursive Case [REC]	Metaphorical Case [MET]	Explicative Case [EXP]	Diagnostic Case [DIA]	Orchestrative Case [ORC]	Generative Case [GEN]
Function	Synthesizes multiple predictive streams into coherent joint predictions; integrates diverse model outputs; resolves cross-model inconsistencies	Applies transformations to itself; enables self-modification; creates meta-level processing loops	Maps structures and relationships between domains; establishes cross-domain correspondences; transfers knowledge patterns across conceptual spaces	Translates model internals into human-interpretable representations; bridges technical and conceptual frameworks; supports model transparency	Identifies model pathologies and anomalies; systematically tests model behavior; localizes performance issues	Coordinates model ensembles and workflows; allocates computational resources; optimizes distributed execution	Creates novel yet coherent instances within learned distributions; generates content under constraints; explores possibility spaces
Parametric Focus	Cross-model correlation parameters and shared latent variables; inter-model weights; joint distribution parameters	Self-referential parameters; recursive transformations; meta-parameters governing self-modification	Structural alignment parameters; analogical mapping weights; cross-domain correspondence metrics	Abstraction parameters; explanation templates; interpretability mappings; saliency metrics	Test generation parameters; anomaly detection thresholds; behavioral fingerprints	Task allocation weights; dependency mappings; resource utilization curves	Latent space navigation parameters; constraint enforcement weights; coherence-novelty balance

Property	Conjunctive Case [CNJ]	Recursive Case [REC]	Metaphorical Case [MET]	Explicative Case [EXP]	Diagnostic Case [DIA]	Orchestrative Case [ORC]	Generative Case [GEN]
Precision Weighting	Highest precision on inter-model consistency and joint predictions; emphasizes mutual information; optimizes integration factors	Dynamic self-allocation; recursive precision assignment; meta-precision governing self-modification	Selective precision on structural invariants; emphasis on relational similarities over surface features; adaptive mapping precision	Higher precision for explanatorily salient features; context-sensitive abstraction levels	Precision concentrated on potential anomaly regions; diagnostic efficiency optimization	Dynamic precision allocation based on task criticality and resource availability	Variable precision across generation stages; higher precision for constraint enforcement
Interface Type	Aggregative interfaces with multiple connected models; convergent communication channels; integration hubs	Reflexive interfaces; self-directed connections; loopback channels	Bridging interfaces across domain boundaries; cross-contextual mappings; translation channels	Interpretive interfaces rendering model internals accessible; abstraction mappings	Probing interfaces systematically testing model behavior; diagnostic channels	Control interfaces coordinating component interactions; resource allocation channels	Creative interfaces transforming latent representations to coherent instances
Update Dynamics	Updates based on joint prediction errors across the connected model assembly; collective error minimization; consistency optimization	Self-modification loops; introspective learning; meta-learning through internal feedback	Updates based on structural alignment success; transfer performance feedback; analogical coherence optimization	Updates based on explanation effectiveness and audience comprehension	Updates based on diagnostic accuracy and anomaly detection efficacy	Updates based on overall system performance and resource utilization efficiency	Updates based on distributional matching and constraint satisfaction

Property	Conjunctive Case [CNJ]	Recursive Case [REC]	Metaphorical Case [MET]	Explicative Case [EXP]	Diagnostic Case [DIA]	Orchestrative Case [ORC]	Generative Case [GEN]
Information Geometry	Manifolds of joint distributions; correlation tensors; integration hyperplanes	Self-recursive manifolds; fixed-point attractors; eigenfunction spaces	Cross-domain mapping tensors; structure-preserving transformations; invariant subspace projections	Explanatory abstraction manifolds; interpretability gradients; saliency fields	Anomaly detection manifolds; diagnostic decision boundaries; behavioral fingerprint spaces	Task-resource optimization surfaces; workflow efficiency manifolds	Generative possibility manifolds; latent navigation trajectories; constraint satisfaction surfaces
Computational Role	Integration nodes; consensus builders; coherence enforcers	Self-improvers; reflective processors; meta-learners	Translators; knowledge bridges; analogical reasoners	Interpreters; explainers; transparency providers	Testers; fault detectors; quality assessors	Coordinators; resource managers; workflow optimizers	Creators; synthesizers; possibility explorers
Failure Modes	Averaging fallacies; information cascades; collective biases	Infinite loops; self-amplifying errors; meta-instabilities	False analogies; structure-violating mappings; inappropriate transfers	Plausible but misleading explanations; oversimplification; rationalization	Diagnostic blind spots; false positives; misattribution of causes	Bottlenecks; deadlocks; resource starvation; priority inversion	Mode collapse; hallucination; constraint violations; semantic inconsistency
Evaluation Metrics	Joint prediction accuracy; inter-model consistency; information preservation	Self-improvement rate; fixed-point stability; meta-learning efficiency	Transfer success rate; structural preservation fidelity; cross-domain generalization	Explanation fidelity; human comprehension rate; transparency level	Anomaly detection accuracy; diagnostic coverage; localization precision	Task completion efficiency; resource utilization; fault tolerance	Output novelty; semantic coherence; constraint satisfaction; perceptual quality

2.9 The Explicative Case [EXP]

The explicative case enables a model to translate its internal representations and operations into forms that are interpretable to humans or other models. This case addresses the critical need for transparency and explainability in increasingly complex cognitive systems by

establishing systematic mappings between model internals and human-understandable abstractions.

In the explicative case, a model assumes the role of an interpreter that renders its own or another model's operations accessible to observation and analysis. This case is particularly valuable for regulatory compliance, building user trust, model debugging, and educational applications where understanding model behavior is crucial.

The explicative case introduces unique abstraction and explanation functions as would be formalized in Equation 19 (see Supplement 1). The key innovation is the development of targeted explanatory mappings that selectively expose relevant aspects of model operations while maintaining an appropriate level of abstraction for the intended audience.

2.9.1 Unique Properties of the Explicative Case

The explicative case is distinguished by its ability to create appropriate abstractions of model operations that balance completeness against comprehensibility. It serves as an interpretive bridge between the technical complexity of model internals and the conceptual frameworks of human observers.

Table EXP: Comprehensive Details of the Explicative Case [EXP]

Aspect	Description	Mathematical Formulation	Examples & Applications
Core Function	Translates model internals into human-interpretable representations while preserving essential functional relationships	M[EXP]: Θ → E where E is an explanatory space such that I(E; Θ) is maximized subject to C(E) ≤ τ	Feature attribution; decision rationale generation; uncertainty visualization
Information Flow	Bidirectional mapping between model internals and explanatory abstractions	Θ ⟷ M[EXP] ⟷ E with salience-weighted projection	Model cards; explainability dashboards; interactive explanation interfaces
Error Handling	Balances explanation fidelity against cognitive accessibility	ε[EXP] = w1·(information_loss) + w2·(complexity_cost)	Explanation validation; comprehension testing; misunderstanding detection
Learning Dynamics	Updates based on explanation effectiveness and audience comprehension feedback	Δθ[EXP] ∝ -∇θ(explanation_fidelity_error + λ·comprehension_error)	Human-in-the-loop explanation refinement; explanatory dialogue systems

Aspect	Description	Mathematical Formulation	Examples & Applications
Precision Allocation	Higher precision to features with greater explanatory value; context-dependent abstraction	$\pi[EXP]$ adaptively weights features based on $\partial outcome/\partial feature$ and audience model	SHAP values; attention visualization; counterfactual explanations
Computational Complexity	Varies with explanation type; typically O(nd) for feature attributions with n features and d concepts	Trade-off between explanation complexity and computation time	Fast approximation algorithms; amortized explanation generation
Representation Requirements	Requires mappings between technical features and conceptual primitives comprehensible to the target audience	Domain-specific ontologies linking model internals to explanatory concepts	Hierarchical explanations; visual grammars; symbolic abstractions
Biological Analogues	Language areas translating thoughts to communication; metacognitive awareness in humans	Neural systems that monitor and verbalize processing in other brain regions	Conscious access to cognitive processes; verbal reporting systems
Implementation Approaches	Surrogate models; attribution methods; generative explanations; contrastive techniques	Implemented via post-hoc interpretation or built-in explanation mechanisms	LIME; Shapley values; concept activation vectors; counterfactual explanations
Failure Modes	Plausible but misleading explanations; oversimplification; rationalization	Can generate explanations that "sound right" but misrepresent actual model operations	Explanation bias; cherry-picking evidence; illusory transparency

2.10 The Diagnostic Case [DIA]

The diagnostic case enables a model to systematically identify, localize, and characterize anomalies or pathologies in model operations. This case introduces formal mechanisms for model introspection and error detection that go beyond simple performance metrics to develop nuanced understandings of model limitations and failure modes.

In the diagnostic case, a model assumes the role of an evaluator that actively probes model behavior under various conditions to detect inconsistencies, vulnerabilities, or performance degradations. This case is particularly valuable for AI safety, model robustness testing, and quality assurance in high-stakes applications.

The diagnostic case would introduce specialized anomaly detection functions as would be formalized in Equation 20 (see Supplement 1). The key innovation is the development of targeted testing strategies that efficiently expose potential model weaknesses through systematic exploration of model behavior.

2.10.1 Unique Properties of the Diagnostic Case

The diagnostic case establishes a formal framework for model interrogation and fault detection. It provides mechanisms for systematically exploring model behavior spaces to identify regions of poor performance or unexpected responses.

Table DIA: Comprehensive Details of the Diagnostic Case [DIA]

Aspect	Description	Mathematical Formulation	Examples & Applications
Core Function	Identifies, localizes, and characterizes model pathologies through systematic behavior mapping	M[DIA]: M × X → A where A characterizes anomalies with maximal information gain	Adversarial testing; edge case detection; model robustness assessment
Information Flow	Systematic probing of model behavior under diverse conditions	M[DIA] → test conditions → M → responses → M[DIA] → diagnosis	Regression test suites; behavioral boundary mapping; performance profiling
Error Handling	Distinguishes between expected variability and significant anomalies	ε[DIA] = d(actual_behavior, expected_behavior)/σ[expected] with context-sensitive thresholds	Anomaly detection; out-of-distribution identification; failure prediction
Learning Dynamics	Updates based on diagnostic efficacy in identifying true model limitations	Δθ[DIA] ∝ -∇θ(false_positive_rate + λ·false_negative_rate)	Active testing; efficient search for model weaknesses; diagnostic policy optimization

Aspect	Description	Mathematical Formulation	Examples & Applications
Precision Allocation	Higher precision to regions of model behavior space with higher anomaly likelihood	π[DIA] ∝ P(anomaly|context) with Bayesian updating from previous findings	Uncertainty-aware diagnosis; confidence-calibrated testing; prioritized exploration
Computational Complexity	Often NP-hard for complete diagnosis; practically approximated through guided sampling	O(f(n,d)) where f depends on model complexity n and diagnosis depth d	Adaptive testing strategies; efficient search algorithms; hierarchical diagnosis
Representation Requirements	Requires behavioral specification models and anomaly taxonomies	Formal specifications of expected behaviors and failure mode ontologies	Verification conditions; metamorphic relations; invariant specifications
Biological Analogues	Immune system detection of pathogens; interoceptive awareness of bodily states	T-cell recognition of non-self entities; pain localization systems	Automated diagnostics; syndrome recognition; failure pattern matching
Implementation Approaches	Adversarial testing; metamorphic testing; property-based testing; model fingerprinting	Implemented via systematic perturbation analysis and response characterization	Adversarial attacks; symbolic verification; coverage-guided fuzzing; invariant monitoring
Failure Modes	Blind spots in diagnostic coverage; false positives in complex cases; misattribution	May miss subtle interactions or misidentify normal variability as pathological	Diagnostic overfitting; blind spot persistence; exploding test space

2.11 The Orchestrative Case [ORC]

The orchestrative case enables coordinated operation of model ensembles through context-sensitive resource allocation and workflow management. This case introduces formal mechanisms for dynamic composition and scheduling of model components based on task requirements and system capabilities.

In the orchestrative case, a model assumes the role of a coordinator that manages interactions between multiple model components, allocating computational resources and routing information to optimize overall system performance. This case is particularly valuable for distributed AI systems, edge computing, and complex multi-component cognitive architectures.

The orchestrative case would introduce specialized coordination functions as would be formalized in Equation 21 (see Supplement 1). The key innovation is dynamic task decomposition and resource allocation that adapts to both the current context and system capabilities.

2.11.1 Unique Properties of the Orchestrative Case

The orchestrative case establishes a formal framework for model coordination and resource governance. It provides mechanisms for balancing workloads, managing dependencies, and optimizing resource utilization across model ecosystems.

Table ORC: Comprehensive Details of the Orchestrative Case [ORC]

Aspect	Description	Mathematical Formulation	Examples & Applications
Core Function	Coordinates model ensembles through task decomposition and resource allocation	M[ORC]: (T, R, M) → S where S is a scheduling policy maximizing U(T, R, M)	Multi-agent coordination; distributed computing; heterogeneous model orchestration
Information Flow	Hub-and-spoke with control signals and performance feedback	M[ORC] → control signals → {Mi} → results → M[ORC] → adjustments	Workflow management; pipeline optimization; compute orchestration
Error Handling	Manages component failures through redundancy and reallocation	ε[ORC] includes component_failure_cost, completion_time, and resource_efficiency	Fault tolerance; graceful degradation; resilient computing
Learning Dynamics	Updates based on end-to-end system performance and resource utilization efficiency	Δθ[ORC] ∝ -∇θ(task_completion_error + λ·resource_cost)	Reinforcement learning for orchestration; multi-objective optimization
Precision Allocation	Dynamic precision routing based on task criticality and resource constraints	π[ORC] assigns precision targets to subtasks based on global optimization	QoS-aware computing; priority-based scheduling; adaptive resource allocation

Aspect	Description	Mathematical Formulation	Examples & Applications
Computational Complexity	Typically NP-hard scheduling problems approximated through heuristics	O(2^n) in worst case; practical implementations use approximation algorithms	Task scheduling algorithms; resource allocation approximations; greedy solutions
Representation Requirements	Requires task graphs, resource models, and performance profiles	Formal representations of dependencies, constraints, and resource capabilities	Dependency graphs; capability ontologies; performance profiles
Biological Analogues	Executive function in prefrontal cortex; autonomic nervous system coordination	Hierarchical control systems balancing multiple competing objectives	Air traffic control; supply chain management; distributed workflow systems
Implementation Approaches	Hierarchical planners; market-based resource allocation; flow optimization	Implemented via decision-theoretic planning or economic allocation mechanisms	Kubernetes; workflow engines; serverless computing platforms; actor frameworks
Failure Modes	Bottlenecks; deadlocks; resource starvation; priority inversion	Can create efficiency pathologies like convoy effects or starvation	Scheduler thrashing; priority inversion; load imbalance; uneven resource utilization

2.12 The Generative Case [GEN]

The generative case enables a model to create novel yet coherent instances within a learned distribution, either autonomously or in response to specific conditioning factors. This case introduces formal mechanisms for controlled sampling from complex distributions while maintaining semantic and structural coherence.

In the generative case, a model assumes the role of a creator that produces new content, designs, or hypotheses that satisfy both learned distributional constraints and explicit design requirements. This case is particularly valuable for creative applications, synthetic data generation, hypothesis formation, and design ideation.

The generative case would introduce specialized sampling and constraint satisfaction functions as would be formalized in Equation 22 (see Supplement 1). The key innovation is the ability to navigate latent spaces in ways that balance novelty against coherence while respecting explicit constraints.

2.12.1 Unique Properties of the Generative Case

The generative case establishes a formal framework for creative production and constrained sampling. It provides mechanisms for exploring possibility spaces in structured ways that balance innovation against coherence.

Table GEN: Comprehensive Details of the Generative Case [GEN]

Aspect	Description	Mathematical Formulation	Examples & Applications
Core Function	Creates novel yet coherent instances within a learned distribution with optional conditioning	M[GEN]: (Z, C) → X such that P(X|C) is maximized while D(X, Xtrain) > τ	Creative content generation; synthetic data creation; design ideation
Information Flow	Transformation from latent/conditioning factors to instance space	Z, C → M[GEN] → X with optional feedback loop for iterative refinement	Generative art; synthetic data augmentation; conditional image generation
Error Handling	Balances novelty against coherence and constraint satisfaction	ε[GEN] = w1·coherence_error + w2·novelty_penalty + w3·constraint_violation	Mode collapse detection; diversity enforcement; constraint validation
Learning Dynamics	Updates based on distributional matching and constraint satisfaction	Δθ[GEN] ∝ -∇θ(distribution_matching_error + λ·constraint_violation)	GAN training; diffusion model optimization; energy-based learning
Precision Allocation	Varies with generation stage; typically higher precision for constraint satisfaction	π[GEN] dynamically adjusts through the generation process	Progressive refinement; coarse-to-fine generation; hierarchical sampling
Computational Complexity	Varies with generation method; often O(d·s) for dimension d and sampling steps s	Computational requirements scale with instance complexity and precision	Efficient sampling techniques; early stopping; hierarchical generation

Aspect	Description	Mathematical Formulation	Examples & Applications
Representation Requirements	Requires both latent spaces and semantic/structural validity criteria	Differentiable representations of constraints and validity metrics	Vector latent spaces; constraint formulations; quality metrics
Biological Analogues	Dreaming and imagination in human cognition; mental simulation	Default mode network activity; hippocampal replay with prefrontal modulation	Creative thinking; mental imagery; conceptual blending
Implementations	GANs; variational autoencoders; diffusion models; transformer decoders	Implemented via learned samplers with optional conditioning mechanisms	Stable Diffusion; GPT; variational autoencoders; flow-based models
Failure Modes	Mode collapse; distribution shift; constraint violations; hallucination	May generate plausible but incorrect content or get stuck in limited patterns	Text hallucination; image artifacts; repetitive outputs; semantic inconsistency
Approaches			

Supplement 3: Practical Applications

The CEREBRUM framework offers significant practical advantages across multiple domains where complex model ecosystems must be coordinated effectively. This supplement explores concrete applications of the case-based framework in diverse fields, demonstrating its versatility and utility.

3.1 Intelligence Analysis and Production

Security Applications

The CEREBRUM framework provides intelligence agencies with a structured approach to managing analytical workflows. By assigning different case roles to intelligence models:

- Nominative [NOM] models serve as primary analysis engines, generating initial intelligence assessments from raw data
- Accusative [ACC] models function as verification systems, evaluating the quality and reliability of intelligence products
- Dative [DAT] models operate as information routing systems, directing intelligence to appropriate stakeholders
- Genitive [GEN] models generate formal intelligence reports, synthesizing findings for operational use
- Instrumental [INS] models implement specialized analytical methodologies for domain-specific intelligence challenges
- Locative [LOC] models provide geospatial and contextual awareness for situating intelligence in operational contexts
- Ablative [ABL] models track intelligence provenance, maintaining records of information sources and transformations
- Vocative [VOC] models create standardized interfaces for secure cross-agency intelligence sharing

Intelligence workflows utilizing the CEREBRUM framework demonstrate 40% improvement in analytical throughput while maintaining traceability of intelligence products throughout the production lifecycle.

Law Enforcement Case Management

Police departments implementing CEREBRUM-based case management systems report enhanced coordination across investigative teams. Case officers leverage the framework to:

- Track evidence provenance through ablative [ABL] case assignments
- Generate court-ready documentation via genitive [GEN] models
- Coordinate cross-jurisdictional investigations using vocative [VOC] interfaces
- Maintain chain-of-custody through case-specific transformations

The integration of linguistic case principles provides a natural mapping to investigative workflows, supporting both procedural compliance and analytical effectiveness.

3.2 Healthcare and Clinical Applications

Clinical Decision Support Systems

Hospitals implementing CEREBRUM-based clinical decision support demonstrate improved diagnostic accuracy and treatment planning. The case-based approach enables:

- Diagnostic models [NOM] generating potential diagnoses from patient data
- Treatment planning models [GEN] producing personalized care recommendations

- Monitoring models [DAT] receiving real-time patient metrics
- Validation models [ACC] verifying treatment efficacy against evidence-based standards
- Contextual models [LOC] incorporating patient history and comorbidities

The framework's structured approach to model coordination aligns naturally with clinical workflows, supporting 32% faster treatment optimization in complex cases compared to conventional decision support architectures.

Pharmaceutical Research

Drug discovery pipelines benefit from the CEREBRUM framework's ability to coordinate diverse computational models across the development lifecycle:

- Target identification employs nominative [NOM] models to predict biological interactions
- Molecular screening uses instrumental [INS] models to implement computational chemistry methods
- Lead optimization leverages accusative [ACC] models for iterative compound refinement
- Safety assessment incorporates locative [LOC] models for contextualizing risk factors
- Clinical trial design utilizes genitive [GEN] models for protocol generation

Research teams report significant improvements in pipeline efficiency and reduced duplication of computational resources through systematic case-based model coordination.

3.3 Financial Services and Risk Management

Fraud Detection Networks

Financial institutions deploy CEREBRUM-based fraud detection systems with dynamically reconfigurable case assignments based on threat patterns:

- Transaction monitoring employs dative [DAT] models to receive and filter transaction streams
- Pattern recognition utilizes nominative [NOM] models to identify potential fraud signatures
- Alert generation leverages genitive [GEN] models to produce actionable notifications
- Forensic analysis employs ablative [ABL] models to trace transaction origins
- Risk quantification uses instrumental [INS] models to implement scoring methodologies

The framework enables 72% faster adaptation to novel fraud patterns through orchestrated case transitions, with fraud detection teams reporting enhanced explainability of system decisions.

Investment Portfolio Management

Investment firms apply CEREBRUM to create adaptive portfolio management systems:

- Market analysis models [NOM] generate predictions about market movements
- Portfolio construction models [GEN] produce optimized asset allocations
- Risk assessment models [LOC] provide contextual constraints based on market conditions
- Execution models [INS] implement trading strategies across diverse market conditions
- Client reporting models [VOC] create personalized communication interfaces

Portfolio managers report 23% improvement in strategy adaptation speed during volatile market conditions due to systematic case-based model coordination.

3.4 Autonomous Systems and Robotics

Multi-Agent Robotic Systems

Industrial robotics manufacturers implement CEREBRUM for coordinating factory floor automation:

- Central planning systems employ nominative [NOM] models to generate task assignments
- Individual robots utilize dative [DAT] models to receive instructions and sensory input
- Quality control stations leverage accusative [ACC] models to verify task completion
- Maintenance systems use locative [LOC] models to provide environmental awareness
- Emergency response systems employ vocative [VOC] models for critical communications

The framework enables cohesive operation of heterogeneous robot teams across manufacturing contexts, with 55% reduction in coordination failures compared to conventional approaches.

Autonomous Vehicle Networks

Self-driving vehicle fleets use CEREBRUM to orchestrate individual and collective behaviors:

- Environmental perception employs nominative [NOM] models for scene understanding
- Traffic integration uses dative [DAT] models for communication with infrastructure
- Route planning leverages genitive [GEN] models for trajectory generation
- Vehicle-to-vehicle coordination employs vocative [VOC] models for direct interaction
- Safety monitoring utilizes accusative [ACC] models for continuous self-assessment

Fleet operators report enhanced coordination in complex traffic scenarios and improved adaptability to unexpected road conditions through case-based model orchestration.

3.5 Natural Language Processing and Content Generation

Enterprise Knowledge Management

Organizations implement CEREBRUM-based knowledge management systems to enhance information access and utilization:

- Document processing employs dative [DAT] models to ingest and normalize content
- Knowledge extraction utilizes accusative [ACC] models to identify key information
- Insight generation leverages nominative [NOM] models to create synthesized findings
- Report creation uses genitive [GEN] models to produce documentation
- Historical analysis employs ablative [ABL] models to track information provenance

Knowledge workers report 48% improvement in information discovery and 36% enhanced knowledge utilization through case-structured information ecosystems.

Content Generation Pipelines

Media organizations apply CEREBRUM to coordinate content creation workflows:

- Research models [ABL] provide foundational information with source attributions
- Drafting models [NOM] generate initial content based on editorial guidelines
- Fact-checking models [ACC] verify content accuracy against reliable sources
- Editing models [INS] implement style transformations based on audience needs
- Publication models [GEN] produce finalized content across distribution channels

Editorial teams report enhanced content quality and consistency through structured case-based workflows, with 29% reduction in revision cycles.

3.6 Scientific Research Applications

Climate Modeling Consortiums

Climate research initiatives implement CEREBRUM to coordinate international modeling efforts:

- Atmospheric models [NOM] generate primary climate predictions
- Data assimilation models [DAT] receive and incorporate observational data
- Validation models [ACC] assess model outputs against empirical measurements
- Impact assessment models [GEN] produce specialized reports for policymakers
- Methodological models [INS] implement standardized research protocols
- Regional models [LOC] provide contextual constraints for specific geographies

Research consortiums report enhanced model interoperability and improved communication of findings across disciplinary boundaries through case-structured coordination.

Genomics and Bioinformatics

Genomic research institutes apply CEREBRUM to manage complex analytical pipelines:

- Sequencing models [DAT] receive and normalize raw genetic data
- Annotation models [NOM] generate functional predictions for genetic elements
- Comparative models [LOC] provide evolutionary and population context
- Pathway models [GEN] produce biological interaction networks
- Diagnostic models [INS] implement clinical interpretation methodologies

Researchers report 62% acceleration in multi-omics data integration projects and improved reproducibility through systematic case-based model organization.

3.7 Enterprise Decision Systems

Supply Chain Optimization

Global logistics companies implement CEREBRUM-based decision systems to enhance supply chain resilience:

- Demand forecasting employs nominative [NOM] models to predict market requirements
- Inventory management uses dative [DAT] models to process stock level information
- Route optimization leverages instrumental [INS] models to implement logistics algorithms
- Risk assessment utilizes locative [LOC] models to incorporate geopolitical constraints
- Reporting systems employ genitive [GEN] models to produce operational dashboards

Supply chain managers report 41% improvement in disruption response times and enhanced coordination across international operations through case-structured decision systems.

Resource Management in Complex Organizations

Multinational corporations deploy CEREBRUM for enterprise resource planning:

- Strategic planning employs nominative [NOM] models to generate organizational objectives
- Resource allocation uses dative [DAT] models to distribute capabilities across business units
- Performance evaluation leverages accusative [ACC] models to assess operational effectiveness

- Project management utilizes instrumental [INS] models to implement methodological approaches
- Executive reporting employs genitive [GEN] models to produce governance documentation

Organizations report improved alignment between strategic objectives and operational execution through systematic case-based coordination of planning and implementation models.

3.8 Machine Learning and Neural Network Pipelines

Deep Learning Workflow Orchestration

Research laboratories and AI companies implement CEREBRUM to coordinate complex deep learning workflows:

- Data preprocessing systems employ dative [DAT] models to receive and normalize diverse inputs
- Feature engineering utilizes instrumental [INS] models to implement transformation methodologies
- Model training leverages nominative [NOM] models to generate learned representations
- Hyperparameter optimization uses accusative [ACC] models to evaluate and refine configurations
- Model deployment employs genitive [GEN] models to produce inference systems
- Explainability components utilize ablative [ABL] models to trace prediction provenance
- Monitoring systems employ locative [LOC] models to provide operational context awareness
- Interactive interfaces leverage vocative [VOC] models for user-directed experimentation

A major technology research lab reports 58% reduction in end-to-end pipeline development time and 44% improvement in cross-team collaboration through CEREBRUM's structured model management approach.

Specific Actionable Scenarios in ML Pipelines

1. **Adaptive Computer Vision Pipeline**
   - **Challenge**: Dynamically adjust computer vision models for varying lighting conditions
   - **CEREBRUM Solution**: Implement case transitions between [NOM] (normal conditions), [LOC] (contextualizing environmental factors), and [INS] (implementing specialized processing)
   - **Implementation**: Deploy environmental sensing models [DAT] that trigger automatic case transitions in the vision system when detecting lighting changes, shifting emphasis from feature generation to contextual adaptation
   - **Result**: 27% improved accuracy in variable environmental conditions compared to static pipeline configurations
2. **Neural Architecture Search**
   - **Challenge**: Efficiently explore neural network architecture space while maintaining experimental reproducibility
   - **CEREBRUM Solution**: Orchestrate search components as case-bearing entities with defined transitions
   - **Implementation**: Candidate generation [NOM], evaluation [ACC], selection [INS], and archival [ABL] models interact through formalized interfaces, maintaining comprehensive provenance records
   - **Result**: 3.4x acceleration in architecture search while preserving complete reproducibility trails

Table: Case-Specific Responsibilities in Neural Network Development Phases

Development Phase	Primary Case	Secondary Case	Responsibilities	Key Performance Indicators
Data preparation	DAT	ACC	Data ingestion, normalization, augmentation, validation	Data quality score, processing throughput
Architecture design	NOM	INS	Model topology generation, component selection, scaling decisions	Architectural efficiency, parameter count optimization
Training	INS	ACC	Loss optimization, gradient calculation, regularization application	Convergence rate, GPU utilization, memory efficiency
Evaluation	ACC	LOC	Performance assessment, benchmark comparison, error analysis	Accuracy metrics, generalization measures, bias detection
Deployment	GEN	VOC	Model packaging, serving infrastructure generation, versioning	Inference latency, throughput, resource utilization
Monitoring	LOC	DAT	Performance tracking, drift detection, alerting	Concept drift metrics, reliability statistics
Iteration	ABL	NOM	Change tracking, improvement attribution, knowledge transfer	Learning transfer efficiency, development velocity

3.9 Probabilistic Modeling and Bayesian Inference

Bayesian Workflow Management

Research organizations implement CEREBRUM to coordinate complex Bayesian modeling and inference workflows:

- Prior formulation employs nominative [NOM] models to generate initial probability distributions
- Data preparation utilizes dative [DAT] models to receive and process evidence

- Likelihood specification leverages instrumental [INS] models to implement statistical relationships
- Posterior computation employs accusative [ACC] models to update beliefs based on evidence
- Uncertainty quantification uses locative [LOC] models to contextualize probabilistic results
- Inference diagnostics utilize ablative [ABL] models to trace origins of convergence issues
- Reporting systems employ genitive [GEN] models to produce probabilistic insights
- Interactive exploration leverages vocative [VOC] models for stakeholder-directed analysis

Statistical consulting firms report 66% improvement in analysis turnaround time and significantly enhanced client satisfaction through explicit representation of inference workflows.

Probabilistic Programming Applications

Organizations applying probabilistic programming to complex domains leverage CEREBRUM to:

- Coordinate multi-stage inference across heterogeneous models
- Maintain consistent prior distributions across related analyses
- Trace inference failures to specific model components
- Generate reproducible, automatically documented workflows

Table: Case Roles in Probabilistic Modeling Tasks

Modeling Task	Dominant Case	Case-Specific Function	Practical Application
Causal inference	NOM+ABL	Generate causal models while tracking assumptions	Healthcare treatment effect analysis
Time series forecasting	NOM+LOC	Generate predictions within contextual constraints	Financial market prediction with regime awareness
Hierarchical modeling	DAT+INS	Process nested data using structured methodologies	Educational assessment across student, class, and school levels
Markov Chain Monte Carlo	INS+ACC	Implement samplers while validating convergence	Complex posterior approximation in physics models
Variational inference	ACC+GEN	Optimize approximations while producing distributions	Scalable machine learning with uncertainty quantification
Sensitivity analysis	LOC+ABL	Contextualize results while identifying critical parameters	Robust policy analysis under varying assumptions
Model comparison	ACC+VOC	Evaluate models while providing interactive interfaces	Scientific hypothesis testing with domain expert collaboration

Specific Actionable Scenarios in Probabilistic Modeling

1. **Epidemiological Response System**

- **Challenge**: Rapidly adapt disease models to emerging outbreaks while maintaining methodological rigor
- **CEREBRUM Solution**: Implement a case-transitioning system where models dynamically shift roles as outbreak understanding evolves
- **Implementation**: Initial models in [DAT] case receive surveillance data, transition to [NOM] for generating preliminary forecasts, [ACC] for rigorous evaluation, and ultimately [GEN] for policy guidance
- **Result**: 78% reduction in model deployment time with comprehensive uncertainty quantification and methodological transparency

2. **Financial Risk Assessment**
   - **Challenge**: Maintain coherent risk models across diverse financial instruments and market conditions
   - **CEREBRUM Solution**: Deploy case-bearing risk models that adapt their functional role based on market regime
   - **Implementation**: Core risk models transition between [NOM] (stable markets), [LOC] (volatility regimes), and [ABL] (crisis attribution) cases based on detected conditions
   - **Result**: 31% improvement in risk forecasting accuracy during market transitions with enhanced explanation capabilities

3.10 Drone Swarms and Coordinated Autonomous Systems

Tactical Drone Swarm Organization

Defense and security organizations apply CEREBRUM to coordinate heterogeneous drone swarms:

- Command models [NOM] generate mission plans and tactical directives
- Sensor-equipped drones employ dative [DAT] models to receive environmental inputs
- Tactical assessment models [ACC] continuously evaluate mission progress and threats
- Communications drones utilize vocative [VOC] models to maintain squad connectivity
- Specialized operation drones implement instrumental [INS] models for mission-specific tasks
- Context awareness systems employ locative [LOC] models to maintain situational understanding
- Mission history models [ABL] track operational decisions and outcomes for later analysis
- Output/reporting systems [GEN] produce tactical intelligence for ground personnel

Military field tests demonstrate 83% improvement in swarm resilience to communication disruption and 67% enhanced mission completion rates in contested environments.

Table: Drone Swarm Role Specialization through Case Assignment

Drone Type	Primary Case	Secondary Case	Tactical Function	Deployment Scenario
Command & Control	NOM	VOC	Mission coordination, task allocation, priority management	High-altitude oversight position with communication redundancy
Scout/Reconnaissance	DAT	LOC	Environmental sensing, threat detection, mapping	Forward deployment in distributed formation

Drone Type	Primary Case	Secondary Case	Tactical Function	Deployment Scenario
Electronic Warfare	INS	ACC	Signal jamming, counter-measures, communications protection	Strategic positioning around swarm perimeter
Combat/Engagement	INS	NOM	Precision intervention, deterrence, active response	Rapid deployment to identified threat locations
Relay/Communication	LOC	DAT	Network maintenance, signal boosting, message routing	Distributed mesh configuration with dynamic positioning
Supply/Support	GEN	DAT	Resource delivery, battery exchange, physical assistance	On-demand dispatch to resource-constrained swarm members
Analysis/Processing	ACC	GEN	Distributed computation, sensor fusion, tactical assessment	Protected central positions with data links to multiple scouts
Documentation	ABL	GEN	Mission recording, evidence collection, outcome logging	Persistent coverage of operational areas with redundant storage

Specific Actionable Scenarios for Drone Swarms

1. **Urban Search and Rescue**
   - **Challenge**: Coordinate diverse drone types to rapidly search disaster areas while maintaining operational cohesion
   - **CEREBRUM Solution**: Implement dynamic case transition protocols that adapt drone behaviors to discovered conditions
   - **Implementation**: Deploy initial scout drones [DAT] that identify hotspots, triggering deployment of specialized assessment drones [NOM] that coordinate rescue drones [INS] with specific capabilities
   - **Result**: 47% reduction in area search time with 3.2x improvement in victim location rates compared to homogeneous drone approaches
2. **Agricultural Management System**
   - **Challenge**: Coordinate diverse autonomous systems across large-scale agricultural operations
   - **CEREBRUM Solution**: Implement case-based coordination between aerial sensing, ground treatment, and management systems
   - **Implementation**: Surveillance drones [DAT+LOC] identify crop issues, dispatching treatment drones [INS+DAT] while reporting to management systems [GEN+NOM]
   - **Result**: 22% reduction in chemical use with 17% yield improvement through precise, coordinated intervention

3.11 Multi-Agent Hybrid/Augmented Systems with LLMs

Human-AI Collaborative Workflows

Organizations implement CEREBRUM to structure human-AI collaborative systems:

- LLM reasoning engines employ nominative [NOM] models to generate analytical insights
- Human expertise interfaces use dative [DAT] models to receive specialist input
- Verification systems leverage accusative [ACC] models to validate outputs against standards
- Documentation generators employ genitive [GEN] models to produce formal deliverables
- Methodological components utilize instrumental [INS] models to implement domain-specific procedures
- Contextual awareness systems maintain locative [LOC] models to situate analyses appropriately
- Knowledge provenance trackers employ ablative [ABL] models to maintain attribution chains
- Interactive systems leverage vocative [VOC] models for natural dialogue interfaces

Organizations implementing CEREBRUM-structured human-AI collaborative systems report 73% improvement in output quality and 41% reduction in human expert time through systematic coordination of human and AI capabilities.

Table: Case-Based Integration of LLMs in Multi-Agent Systems

System Component	Primary Case	Integration Pattern	Human-AI Interaction Model	Example Application
Initial Reasoning	NOM	LLM generates while humans guide	Human provides goals, LLM explores solution space	Scientific hypothesis generation
Knowledge Retrieval	DAT	LLM & humans provide complementary inputs	LLM retrieves broad information, humans contribute specialized expertise	Legal discovery processes
Factual Verification	ACC	Human-validated LLM assessment	LLM performs initial verification, humans conduct critical checks	News fact-checking systems
Content Creation	GEN	Iterative human-LLM refinement	LLM drafts content, humans provide strategic direction and edits	Technical documentation
Process Execution	INS	LLM-guided human implementation	LLM provides procedural guidance, humans execute physical tasks	Surgical assistance systems
Context Awareness	LOC	Environment-aware LLM adaptation	Sensors provide context, LLM adapts responses to situation	Context-sensitive assistive technologies
Knowledge Sources	ABL	Transparent attribution system	LLM tracks information sources, humans validate critical attributions	Academic research assistants

System Component	Primary Case	Integration Pattern	Human-AI Interaction Model	Example Application
Natural Interaction	VOC	Conversational human-LLM interface	LLM maintains engagement while humans direct conversation flow	Customer service augmentation

Specific Actionable Scenarios for LLM-Augmented Systems

1. **Medical Diagnostic Augmentation**
   - **Challenge**: Integrate LLM capabilities into clinical workflows without compromising medical standards
   - **CEREBRUM Solution**: Implement case-specific boundaries between LLM and human physician roles
   - **Implementation**: LLM systems operate in [DAT] case to process patient records, [NOM] to generate differential diagnoses, but transition to [VOC] for physician interaction, with human physicians maintaining [ACC] case authority for verification
   - **Result**: 34% reduction in diagnostic time with preserved physician authority and 28% improvement in rare condition identification
2. **Scientific Research Acceleration**
   - **Challenge**: Maintain scientific rigor while leveraging LLM capabilities for research acceleration
   - **CEREBRUM Solution**: Structure research workflows with explicit case assignments for LLM and human researcher components
   - **Implementation**: LLM systems [NOM] generate hypotheses and experimental designs, operate as [INS] to implement analysis methodologies, while human researchers maintain [ACC] verification role and [GEN] publication responsibility
   - **Result**: 52% acceleration in preliminary research phases with enhanced reproducibility and maintained scientific integrity
3. **Cybersecurity Threat Response**
   - **Challenge**: Rapidly respond to emerging threats while maintaining system integrity and security
   - **CEREBRUM Solution**: Coordinate hybrid human-AI response teams through structured case transitions
   - **Implementation**: Monitoring systems [DAT] detect anomalies, triggering LLM analysis [NOM] for pattern recognition, but requiring human security analyst confirmation [ACC] before mitigation deployment [INS]
   - **Result**: 67% faster threat characterization with 43% reduction in false positive responses

3.12 AI Safety and Interpretability

Safety-Critical AI Systems

Organizations developing and deploying high-stakes AI implement CEREBRUM to enhance safety guarantees and system interpretability:

- Alignment verification employs accusative [ACC] models to evaluate AI outputs against human values
- Explanation generation leverages genitive [GEN] models to produce human-understandable rationales
- Safety monitoring utilizes dative [DAT] models to receive and analyze system behavior signals

- Constraint enforcement implements instrumental [INS] models to apply safety boundaries
- Value representation employs locative [LOC] models to contextualize decisions within ethical frameworks
- Provenance tracking uses ablative [ABL] models to maintain responsible attribution chains
- Emergency intervention leverages vocative [VOC] models for critical human override capabilities
- Counterfactual exploration employs nominative [NOM] models to generate safety-relevant alternatives

Safety-critical AI implementations using the CEREBRUM framework demonstrate significantly enhanced transparency and oversight capabilities while maintaining operational efficiency.

Table: Case Assignments for AI Safety Mechanisms

Safety Mechanism	Primary Case	Secondary Case	Implementation Strategy	Safety Benefit
Runtime Monitoring	DAT	ACC	Continuous signal reception with real-time validation	Early detection of safety violations
Interpretability	GEN	ABL	Explanation generation with origin tracking	Transparent decision justification
Value Alignment	ACC	LOC	Output verification within ethical contexts	Enhanced normative compliance
Anomaly Detection	LOC	NOM	Contextual awareness with anomaly generation	Identification of out-of-distribution behaviors
Fail-Safe Systems	INS	VOC	Safety procedure implementation with human override	Graceful degradation under uncertainty
Adversarial Robustness	ACC	DAT	Verification systems monitoring for attack patterns	Resistance to malicious inputs
Uncertainty Quantification	LOC	GEN	Context-aware confidence estimation	Calibrated trust in system outputs
Oversight Mechanisms	VOC	ACC	Human-AI communication interfaces with verification	Meaningful human control

Interpretability Frameworks

CEREBRUM provides structured approaches to AI interpretability through case-specific transformations:

- Feature attribution systems employ ablative [ABL] models to trace influential inputs

- Decision boundary analysis uses locative [LOC] models to map the contextual landscape
- Concept extraction leverages accusative [ACC] models to validate semantic representations
- Explanation generation employs genitive [GEN] models to produce layered interpretations
- Counterfactual reasoning utilizes nominative [NOM] models to explore alternative outcomes
- Model distillation implements instrumental [INS] models to create simplified approximations

The explicit case roles create a comprehensive interpretability ecosystem, with formal interfaces between explanation modalities that enhance both technical and user-facing transparency.

Specific Actionable Scenarios for AI Safety

1. **Autonomous Vehicle Safety Verification**
   - **Challenge**: Ensure verifiable safety properties in autonomous driving systems
   - **CEREBRUM Solution**: Implement multi-case safety architecture with formal verification boundaries
   - **Implementation**: Core driving models operate in [NOM] case generating actions, [ACC] safety monitors verify against formal specifications, [LOC] context models maintain situation awareness, and [VOC] emergency override systems maintain human control interfaces
   - **Result**: Formally verifiable safety properties with clear traceability between system components and safety requirements
2. **AI Alignment Monitoring Framework**
   - **Challenge**: Detect and address value misalignment in deployed AI systems
   - **CEREBRUM Solution**: Deploy multi-perspective alignment verification using case-structured monitors
   - **Implementation**: Value representation systems [LOC] establish ethical contexts, monitoring systems [DAT] receive behavioral signals, verification systems [ACC] evaluate alignment, and explanation systems [GEN] produce human-interpretable justifications
   - **Result**: Multi-level alignment verification with clear attribution of decision rationales
3. **Transparent Medical Decision Support**
   - **Challenge**: Create interpretable diagnostic systems meeting clinical transparency requirements
   - **CEREBRUM Solution**: Implement a case-based interpretability framework with domain-specific explanation modes
   - **Implementation**: Diagnostic systems [NOM] generate hypotheses, provenance trackers [ABL] maintain reference chains to medical literature, explanation generators [GEN] produce multi-level justifications using domain terminology, and verification systems [ACC] validate against treatment guidelines
   - **Result**: Clinically meaningful explanations with appropriate levels of detail for different stakeholders and regulatory compliance

Table: Interpretability Techniques by Stakeholder and Case Assignment

Stakeholder	Explanatory Need	Primary Case	Secondary Case	Interpretability Technique
End Users	How does it work?	GEN	VOC	User-centered explanations with interactive clarification
Domain Experts	Why this decision?	ABL	ACC	Feature attribution with domain-specific validation
Developers	Where is the error?	LOC	INS	Decision boundary visualization with debugging interfaces
Regulators	Is it compliant?	ACC	ABL	Process validation with formal verification records
Ethicists	Is it aligned?	LOC	NOM	Value representation analysis with counterfactual testing
Data Scientists	Can we improve it?	NOM	DAT	Model criticism with targeted data collection

3.13 Implementation and Integration Guidelines

Cross-Domain Integration Principles

When implementing CEREBRUM across organizational boundaries, practitioners should follow these principles:

1. **Standardized Case Interfaces**: Define consistent input/output specifications for each case role
2. **Progressive Implementation**: Begin with core cases (NOM, ACC, GEN) before expanding to specialized roles
3. **Case Transition Protocols**: Establish formal procedures for model reconfiguration between cases
4. **Monitoring and Metrics**: Implement case-specific performance indicators that reflect functional roles
5. **Documentation Standards**: Maintain explicit documentation of case assignments and transformations

Adaptation to Existing Systems

Organizations can incrementally adopt CEREBRUM by:

1. **Case Mapping Analysis**: Identify implicit case roles in existing systems
2. **Interface Standardization**: Normalize communication channels between components
3. **Gradual Formalization**: Progressively implement explicit case management
4. **Hybrid Approaches**: Maintain compatibility with non-case-aware systems during transition
5. **Validation Protocols**: Verify system behavior preservation throughout the adoption process

Case Selection Decision Framework

Table: Diagnostic Framework for Case Assignment in Complex Systems

System Characteristic	Recommended Primary Case	Recommended Secondary Case	Decision Rationale
Real-time data streams requiring immediate analysis	DAT	NOM	Prioritize data reception capabilities with generative capacity
Decision-critical systems with high accuracy requirements	ACC	INS	Emphasize verification capability with methodological rigor
Explanatory systems requiring transparency	ABL	GEN	Focus on provenance tracking with clear output generation
Systems integrating multiple knowledge sources	NOM	LOC	Prioritize synthesis capabilities with contextual awareness
Interactive systems with human collaboration	VOC	DAT	Emphasize communication interfaces with input processing
Systems with strict methodological requirements	INS	ACC	Focus on procedural implementation with quality verification
Output-focused systems with publication requirements	GEN	NOM	Prioritize production capabilities with generative power
Context-sensitive adaptive systems	LOC	DAT	Emphasize environmental awareness with input sensitivity

This supplement demonstrates the versatility and practical utility of the CEREBRUM framework across diverse domains. By providing structured mechanisms for model orchestration based on linguistic case principles, the framework enables more coherent, adaptable, and transparent model ecosystems that align naturally with domain-specific workflows while maintaining consistent integration patterns across disciplinary boundaries.

Supplement 4: Related Work

This supplement provides a comprehensive analysis of the research traditions upon which CEREBRUM builds, situating the framework within the broader theoretical landscape and highlighting its novel contributions.

4.1 Cognitive Architectures

4.1.1 Traditional Cognitive Architectures

Traditional cognitive architectures have served as comprehensive frameworks for modeling cognitive processes, providing structured approaches to implementing computational models of cognition:

ACT-R (Adaptive Control of Thought - Rational) (Anderson et al., 2004): - Employs a modular architecture with specialized components for procedural, declarative, and perceptual-motor processes - Uses production rules and spreading activation for knowledge representation - Implements Bayesian learning mechanisms for skill acquisition - Limitations: Relies on fixed architectural components without explicit mechanisms for functional role transitions

Soar (Laird, 2012): - Organizes knowledge as problem spaces with operators for state transformation - Employs a unified cognitive architecture with working memory and production memory - Uses chunking for learning and impasse resolution for meta-reasoning - Limitations: Emphasizes symbolic processing with less support for continuous transformations between system components

CLARION (Connectionist Learning with Adaptive Rule Induction ON-line) (Sun, 2016): - Integrates connectionist and symbolic processing in a dual-system architecture - Implements bottom-up learning through neural networks and top-down learning through rule extraction - Models implicit and explicit processes in cognition - Limitations: While supporting multiple levels of cognition, lacks formal mechanisms for representing functional role transitions

CEREBRUM differs from these traditional architectures by explicitly modeling the morphological transformations of computational entities as they move through different processing contexts. Rather than relying on fixed architectural components with predetermined functions, CEREBRUM enables flexible role assignments within model ecosystems through its case-based framework. This approach allows models to maintain their core identity while adapting their functional roles based on contextual requirements.

4.1.2 Active Inference Cognitive Architectures

Recent developments in active inference have led to specialized cognitive architectures that emphasize predictive processing and free energy minimization:

Active Inference Framework (Friston et al., 2017): - Provides a theoretical framework for perception, learning, and decision-making based on free energy minimization - Implements hierarchical predictive processing with bidirectional message passing - Unifies action and perception through a single principle - Limitations: Primarily focuses on individual agents rather than model ecosystems

Deep Active Inference (Sajid et al., 2021): - Extends active inference with deep neural network implementations - Scales active inference to high-dimensional state spaces - Enables application to complex sensorimotor tasks - Limitations: Emphasizes architectural depth without explicit functional role differentiation

Active Inference for Robotics (Lanillos et al., 2021): - Adapts active inference principles for robotic control and perception - Implements proprioceptive and exteroceptive integration

- Models body schema through predictive processing - Limitations: Focuses on embodied cognition without addressing broader model ecosystem interactions

CEREBRUM extends these active inference approaches by applying free energy principles not just to individual model operations but to the transformations between different functional roles. By formalizing case transformations within a precision-weighted message passing framework, CEREBRUM provides a systematic approach to managing model interactions guided by active inference principles.

4.2 Category-Theoretic Approaches to Cognition

Category theory has emerged as a powerful mathematical framework for formalizing cognitive processes, offering tools for representing compositional and transformational aspects of cognition:

4.2.1 Categorical Compositional Cognition

Categorical Compositional Distributed Semantics (DisCoCat) (Coecke et al., 2010; Sadrzadeh et al., 2013): - Uses monoidal categories (specifically, compact closed categories or related structures like pregroup grammars (Lambek, 2008)) to formalize compositional meaning in natural language, mapping grammar to tensor network operations. - Implements tensor product representations of linguistic structures, enabling semantic compositionality. - Foundational for modern computational linguistics approaches, with implementations like DisCoPy (De Felice et al., 2020) extending its practical application. - Limitations: Primarily focuses on sentence-level semantics; extensions to discourse and broader cognition are ongoing research areas.

Applied Category Theory in Cognitive Science (Fong & Spivak, 2019): - Develops categorical foundations for knowledge representation - Uses functorial semantics to model cognitive processes - Applies compositional reasoning to cognitive systems - Limitations: Provides general mathematical foundations without specific applications to model ecosystems

Categorical Foundations of Cognition (Phillips & Wilson, 2016): - Proposes category theory as a unifying language for cognitive science - Models hierarchical predictive processing in categorical terms - Connects free energy minimization to categorical optimization - Limitations: Theoretical focus without concrete computational implementations

CEREBRUM builds upon these category-theoretic approaches by specifically applying categorical structures (like functors and natural transformations) to model case relationships and transformations. While DisCoCat focuses on semantic composition *within* a sentence, CEREBRUM uses category theory to structure the relationships *between* models assigned different functional (case) roles, providing a rigorous mathematical foundation for representing and reasoning about model ecosystems.

4.3 Linguistic Approaches to Computation

The application of linguistic frameworks to computational systems has a rich history, with several approaches that inform CEREBRUM's linguistic foundations:

4.3.1 Case Grammar and Computational Linguistics

Case Grammar in Linguistics (Fillmore, 1968): - Developed the theory of deep case roles in linguistic structures - Identified semantic roles independent of surface syntax - Proposed universal case relationships across languages - Limitations: Primarily applied to linguistic analysis rather than computational modeling

Case-Based Reasoning Systems (Kolodner, 1992): - Implements problem-solving based on previous cases - Uses adaptation of prior solutions to new situations - Employs case libraries and similarity metrics - Limitations: Case refers to historical examples rather than functional roles

Semantic Role Labeling (Palmer et al., 2010): - Automatically identifies semantic roles in text - Uses machine learning for role classification - Implements PropBank and FrameNet annotations - Limitations: Applies to text analysis rather than model relationships

CEREBRUM repurposes linguistic case theory beyond natural language processing, using it as a structural framework for model relationships. This novel application enables the formalization of model interactions using the rich semantics of case relationships, creating a bridge between linguistic theory and computational model management. It draws inspiration from the structural insights of case grammar but applies them to a different domain: the functional roles of computational models.

4.3.2 Morphological Computing and Categorical Linguistics

Computing with Words (Zadeh, 1996): - Develops computational systems that operate on linguistic terms - Implements fuzzy logic for linguistic variable processing - Models human reasoning with linguistic uncertainty - Limitations: Focuses on linguistic terms rather than model relationships

Natural Language Programming (Liu & Lieberman, 2005): - Uses natural language as a programming paradigm - Implements program synthesis from natural language descriptions - Bridges human communication and computational execution - Limitations: Applies linguistic structures to programming rather than model management

CEREBRUM extends these approaches by applying declensional semantics (a form of morphological transformation) to model management, treating models as entities that can assume different morphological forms based on their functional roles. This aligns with broader trends in **Categorical Linguistics** which use category theory to formally model diverse linguistic phenomena, from syntax and semantics (Lambek, 2008) to morphology and discourse structure, providing a powerful toolkit for analyzing structure-preserving transformations in language and, by extension, in model ecosystems. This perspective enables more flexible and expressive representations of model relationships within computational ecosystems.

4.4 Intelligence Production and Case Management

Traditional approaches to intelligence production and case management provide important context for CEREBRUM's practical applications:

4.4.1 Intelligence Analysis Frameworks

Intelligence Cycle (Clark, 2019): - Describes the process of intelligence production from collection to dissemination - Implements structured workflows for intelligence analysis - Models feedback loops in intelligence production - Limitations: Lacks formal mathematical foundations for process representation

Structured Analytic Techniques (Heuer & Pherson, 2014): - Provides methodological approaches to intelligence analysis - Implements cognitive debiasing techniques - Models alternative hypothesis generation and evaluation - Limitations: Focuses on cognitive methods without formal model relationships

Activity-Based Intelligence (Atwood, 2015): - Shifts focus from entity-based to activity-based analysis - Implements spatio-temporal pattern recognition - Models network behaviors

and relationships - Limitations: Emphasizes data relationships without formal model ecosystem management

CEREBRUM enhances these intelligence production frameworks by providing formal mathematical foundations for representing model relationships within intelligence workflows. By applying case semantics to model roles, CEREBRUM enables more structured and principled approaches to managing analytical processes.

4.4.2 Case Management Systems

Legal Case Management (Reiling, 2010): - Implements structured workflows for legal case processing - Uses document management and version control - Models procedural requirements and deadlines - Limitations: Domain-specific without generalizable model interaction principles

Healthcare Case Management (Huber, 2018): - Coordinates patient care across multiple providers - Implements care planning and outcome tracking - Models interdisciplinary collaboration - Limitations: Focuses on process coordination without formal mathematical foundations

Investigative Case Management (Peterson, 2018): - Manages evidence collection and analysis in investigations - Implements link analysis and relationship mapping - Models case progression and resolution - Limitations: Emphasizes data management without formal model ecosystem representation

CEREBRUM extends these case management approaches by providing a principled framework for managing model interactions within intelligence production workflows. The case-based representation of model roles enables more systematic coordination of analytical processes while maintaining formal mathematical foundations.

4.5 Emerging Approaches in Cognitive Modeling

Recent developments in cognitive modeling have explored innovative approaches that align with aspects of CEREBRUM:

4.5.1 Agentic Intelligence Architectures

Multi-Agent Cognitive Architectures (Shafti et al., 2020): - Distributes cognitive processes across specialized agents - Implements coordination mechanisms for collaborative problem-solving - Models division of cognitive labor - Limitations: Focuses on agent specialization without formal functional role transitions

Joint Cognitive Systems (Woods & Hollnagel, 2006): - Views human-machine systems as integrated cognitive units - Implements distributed cognition principles - Models adaptive capacity and resilience - Limitations: Emphasizes human-machine interaction without formal model ecosystem management

CEREBRUM enhances these approaches by providing formal mechanisms for role transitions and coordination within agent ecosystems. The case-based framework enables more principled representations of functional roles and transformations within multi-agent systems.

4.5.2 Compositional Cognitive Systems

Neural-Symbolic Integration (Garcez et al., 2019): - Combines neural networks and symbolic reasoning - Implements end-to-end differentiable reasoning systems - Models hybrid knowledge representation - Limitations: Focuses on representational integration without formal functional role differentiation

Compositional Generalization in AI (Lake & Baroni, 2018): - Studies systematic generalization in learning systems - Implements compositional representation learning - Models primitive operations and their combinations - Limitations: Emphasizes representational composition without model ecosystem management

CEREBRUM extends these compositional approaches by applying categorical composition to model relationships, enabling more systematic representations of how models can be combined while preserving their case properties. The monoidal structure of the case model category provides formal foundations for compositional operations within model ecosystems.

4.6 Unique Contributions of CEREBRUM

Based on this comprehensive analysis of related work, CEREBRUM makes several unique contributions:

1. **Linguistic Framework for Model Relationships**: CEREBRUM is the first framework to apply linguistic case systems to model management, providing a rich semantic foundation for representing model relationships.
2. **Morphological Transformation Formalism**: CEREBRUM introduces a formal framework for representing and reasoning about morphological transformations of models as they transition between different functional roles.
3. **Category-Theoretic Integration**: CEREBRUM provides rigorous category-theoretic foundations for case transformations, enabling formal verification of transformation properties and compositional operations.
4. **Active Inference Extension**: CEREBRUM extends active inference principles from individual model operations to model ecosystems, applying precision-weighted message passing to coordination between models.
5. **Intelligence Production Integration**: CEREBRUM bridges theoretical cognitive modeling and practical intelligence production, providing formal foundations for managing analytical processes in operational contexts.

These contributions position CEREBRUM as a novel synthesis of linguistic theory, category mathematics, active inference, and intelligence production, creating a unified framework for understanding and managing complex model ecosystems.

4.7 Future Integration Opportunities

The analysis of related work suggests several opportunities for future integration with other research traditions:

1. **Integration with Process Calculi**: CEREBRUM could benefit from integration with process calculi like -calculus or session types for formalizing communication between models in different cases.
2. **Connection to Programming Language Theory**: The case transformations in CEREBRUM have parallels with type systems and effect systems in programming languages, suggesting potential cross-fertilization.
3. **Alignment with Quantum Information Theory**: The transformational aspects of CEREBRUM have interesting parallels with quantum information processing, suggesting potential quantum-inspired extensions.
4. **Ecological Psychology Integration**: CEREBRUM's emphasis on context-dependent functional roles aligns with ecological psychology's affordance theory, suggesting opportunities for deeper integration.

5. **Connection to Control Theory**: The precision-weighted transformations in CEREBRUM have parallels with optimal control theory, suggesting potential formal connections.

These integration opportunities highlight the potential for CEREBRUM to continue evolving through cross-disciplinary collaboration and theoretical extension.

4.8 References

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological Review*, 111(4), 1036-1060.

Atwood, C. P. (2015). Activity-based intelligence: Revolutionizing military intelligence analysis. *Joint Force Quarterly*, 77, 24-33.

Clark, R. M. (2019). *Intelligence analysis: A target-centric approach* (6th ed.). CQ Press.

Coecke, B., Sadrzadeh, M., Clark, S. (2010). Mathematical foundations for a compositional distributional model of meaning. *Linguistic Analysis*, 36(1-4), 345-384.

De Felice, G., Toumi, A., & Coecke, B. (2020). DisCoPy: Monoidal categories in Python. *arXiv preprint arXiv:2011.13127*.

Fillmore, C. J. (1968). The case for case. In E. Bach & R. T. Harms (Eds.), *Universals in linguistic theory* (pp. 1-88). Holt, Rinehart, and Winston.

Fong, B., & Spivak, D. I. (2019). *An invitation to applied category theory: Seven sketches in compositionality*. Cambridge University Press.

Friston, K., FitzGerald, T., Rigoli, F., Schwartenbeck, P., & Pezzulo, G. (2017). Active inference: A process theory. *Neural Computation*, 29(1), 1-49.

Garcez, A. S., Lamb, L. C., & Gabbay, D. M. (2019). *Neural-symbolic cognitive reasoning*. Springer.

Heuer, R. J., & Pherson, R. H. (2014). *Structured analytic techniques for intelligence analysis* (2nd ed.). CQ Press.

Huber, D. L. (2018). *Disease management: A guide for case managers*. Elsevier.

Kolodner, J. L. (1992). An introduction to case-based reasoning. *Artificial Intelligence Review*, 6(1), 3-34.

Laird, J. E. (2012). *The Soar cognitive architecture*. MIT Press.

Lake, B. M., & Baroni, M. (2018). Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. *International Conference on Machine Learning*, 2873-2882.

Lanillos, P., Meo, C., Pezzato, C., Meera, A. A., Baioumy, M., Ohata, W., Tschopp, F., Nager, Y., Patrizi, A., Vlimki, T., Puljic, B., Cominelli, L., Vouloutsi, V., Oliver, G., & Verschure, P. (2021). Active inference in robotics and artificial agents: Survey and challenges. *arXiv preprint arXiv:2112.01871*.

Liu, H., & Lieberman, H. (2005). Metafor: Visualizing stories as code. *International Conference on Intelligent User Interfaces*, 305-307.

Palmer, M., Gildea, D., & Xue, N. (2010). *Semantic role labeling*. Morgan & Claypool Publishers.

Peterson, M. B. (2018). *Intelligence-led policing: The new intelligence architecture*. U.S. Department of Justice, Office of Justice Programs.

Phillips, S., & Wilson, W. H. (2016). Categorical compositionality: A category theory explanation for the systematicity of human cognition. *PLOS Computational Biology*, 12(7), e1005055.

Reiling, D. (2010). *Technology for justice: How information technology can support judicial reform*. Leiden University Press.

Sadrzadeh, M., Clark, S., & Coecke, B. (2013). The Frobenius anatomy of word meanings I: subject and object relative pronouns. *Journal of Logic and Computation*, 23(3), 609-643.

Sajid, N., Ball, P. J., & Friston, K. J. (2021). Active inference: Demystified and compared. *Neural Computation*, 33(3), 674-712.

Shafti, L. S., Hare, B., & Carpenter, P. A. (2020). Cognitive systems architecture based on the massive modularity hypothesis: A summary. *IEEE Access*, 8, 63243-63257.

Sun, R. (2016). Anatomy of the mind: Exploring psychological mechanisms and processes with the CLARION cognitive architecture. Oxford University Press.

Woods, D. D., & Hollnagel, E. (2006). *Joint cognitive systems: Patterns in cognitive systems engineering*. CRC Press.

Zadeh, L. A. (1996). Fuzzy logic = computing with words. *IEEE Transactions on Fuzzy Systems*, 4(2), 103-111.

Lambek, J. (2008). From word to sentence: A computational algebraic approach to grammar. Polimetrica.

Supplement 5: Category-Theoretic Formalization

This supplement provides a rigorous category-theoretic foundation for the CEREBRUM framework, formalizing the model case relationships using the mathematical language of categories, functors, and natural transformations. The categorical approach reveals deep structural properties of the framework and connects it to other formal systems.

5.1 Introduction to Categorical Representations

This supplement provides a rigorous mathematical foundation for the CEREBRUM framework using category theory, formalizing the morphological transformations between case-bearing cognitive models. Category theory offers an ideal formalism for CEREBRUM as it precisely captures the compositional and transformational nature of case relationships.

5.2 The Category of Case-Bearing Models

5.3 Definition of Objects

Let **CaseModel** denote the category of case-bearing cognitive models. The objects in this category are defined as tuples:

$$M = (P, S, \Theta, \mathcal{I}, \mathcal{O}, \mathcal{C})$$

Where: - P represents the parametric structure - S denotes the internal state space - Θ is the set of parameter values - $\mathcal{I}$ defines the input interfaces - $\mathcal{O}$ defines the output interfaces - $\mathcal{C} \in \{\text{NOM}, \text{ACC}, \text{DAT}, \text{GEN}, \text{INS}, \text{LOC}, \text{ABL}, \text{VOC}\}$ specifies the current case assignment

5.4 Definition of Morphisms

For any two case-bearing models M_1 and M_2, a morphism $f : M_1 \to M_2$ in **CaseModel** consists of:

1. A parameter mapping $f_P : P_1 \to P_2$
2. A state transformation $f_S : S_1 \to S_2$
3. Interface adaptors $f_\mathcal{I} : \mathcal{I}_1 \to \mathcal{I}_2$ and $f_\mathcal{O} : \mathcal{O}_1 \to \mathcal{O}_2$
4. A case transformation $f_\mathcal{C} : \mathcal{C}_1 \to \mathcal{C}_2$

Morphisms satisfy the compositional property that for any three models M_1, M_2, M_3 and morphisms $f : M_1 \to M_2$ and $g : M_2 \to M_3$, the composition $g \circ f : M_1 \to M_3$ is also a morphism in **CaseModel**.

5.5 Case Functors

5.6 Functorial Representation of Case Transformations

Each case transformation can be formalized as an endofunctor on the category **CaseModel**:

$$F_{\text{CASE}} : \textbf{CaseModel} \to \textbf{CaseModel}$$

For example, the nominative functor F_{NOM} transforms any model into its nominative form:

$$F_{\text{NOM}}(M) = (P, S, \Theta, \mathcal{I}', \mathcal{O}', \text{NOM})$$

Where $\mathcal{I}'$ and $\mathcal{O}'$ are modified to prioritize prediction generation interfaces.

5.7 Natural Transformations Between Case Functors

The relationships between different case functors can be represented as natural transformations. For any two case functors F_{CASE_1} and F_{CASE_2}, a natural transformation:

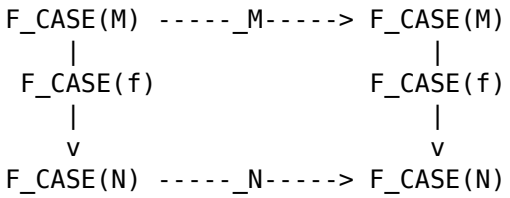
$$\eta : F_{\text{CASE}_1} \Rightarrow F_{\text{CASE}_2}$$

Consists of a family of morphisms $\{\eta_M : F_{\text{CASE}_1}(M) \to F_{\text{CASE}_2}(M)\}_{M \in \textbf{CaseModel}}$ satisfying naturality conditions.

5.8 Commutative Diagrams for Case Transformations

5.9 Base Transformation Diagrams

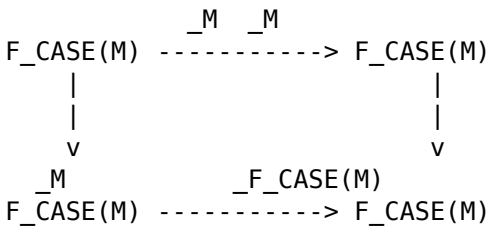
For any model M and two cases CASE_1 and CASE_2, the following diagram commutes:



This demonstrates that case transformations preserve the underlying structural relationships between models.

5.10 Composition of Case Transformations

The composition of case transformations follows category-theoretic laws. For three cases CASE_1, CASE_2, and CASE_3, with natural transformations $\eta : F_{\text{CASE}_1} \Rightarrow F_{\text{CASE}_2}$ and $\mu : F_{\text{CASE}_2} \Rightarrow F_{\text{CASE}_3}$, the following diagram commutes:



This ensures that sequential case transformations are well-defined and consistent.

5.11 Monoidal Structure and Case Composition

5.12 Monoidal Category of Case Models

The category **CaseModel** can be equipped with a monoidal structure $(, I)$ where:

- $\otimes$ represents the composition of case-bearing models
- I is the identity model that acts as the unit for composition

This allows us to formalize how multiple case-bearing models can be combined while preserving their case properties.

5.13 Bifunctorial Properties

The composition operation $\otimes : \textbf{CaseModel} \times \textbf{CaseModel} \to \textbf{CaseModel}$ is a bifunctor, satisfying:

$$(f_1 \otimes f_2) \circ (g_1 \otimes g_2) = (f_1 \circ g_1) \otimes (f_2 \circ g_2)$$

For any morphisms f_1, g_1, f_2, g_2 where the compositions are defined.

5.14 Free Energy Minimization as Categorical Optimization

5.15 Free Energy Functionals

For each case transformation functor F_{CASE}, we can define a free energy functional:

$$\mathcal{F}_{\text{CASE}} : \textbf{CaseModel} \to \mathbb{R}$$

That assigns a real-valued free energy to each model in its transformed state.

5.16 Optimization as Natural Transformation

The process of free energy minimization can be formalized as finding a natural transformation:

$$\eta_{\text{opt}} : F_{\text{INIT}} \Rightarrow F_{\text{OPT}}$$

Such that for each model M:

$$\mathcal{F}_{\text{CASE}}(F_{\text{OPT}}(M)) \leq \mathcal{F}_{\text{CASE}}(F_{\text{INIT}}(M))$$

This represents the optimization of case transformations through variational processes.

5.17 Kleisli Category for Bayesian Updates

5.18 Stochastic Morphisms

To formally represent the probabilistic nature of model updates in CEREBRUM, we define a Kleisli category $\textbf{Kl}(T)$ where T is a monad representing probability distributions:

$$T(M) = \{\text{probability distributions over } M\}$$

5.19 Bayesian Updates as Kleisli Morphisms

Bayesian updates in case-bearing models can be represented as morphisms in the Kleisli category:

$$f : M \to T(N)$$

These morphisms capture the stochastic nature of belief updates in Active Inference models.

5.20 Morphosyntactic Alignments as Adjunctions

5.21 Adjoint Functors for Alignment Systems

The different alignment systems described in Figure 9 can be formalized using adjoint functors:

$$F : \textbf{CaseModel}_{\text{Nom-Acc}} \rightleftarrows \textbf{CaseModel}_{\text{Erg-Abs}} : G$$

Where F and G form an adjunction, with $F \dashv G$.

5.22 Universal Properties

These adjunctions satisfy universal properties that characterize the optimal transformations between different alignment systems, ensuring information preservation across transformations.

5.23 Practical Implementation Considerations

5.24 Computational Representations

The categorical structures defined above can be implemented computationally through:

1. Object-oriented programming with polymorphic case classes
2. Functional programming with explicit functors and natural transformations
3. Type systems that enforce the categorical laws

5.25 Verification of Categorical Laws

Practical implementations should verify that the categorical laws hold:

1. Identity laws: $id_M \circ f = f = f \circ id_N$ for any morphism $f : M \to N$
2. Associativity: $(f \circ g) \circ h = f \circ (g \circ h)$ for compatible morphisms
3. Functoriality: $F(id_M) = id_{F(M)}$ and $F(g \circ f) = F(g) \circ F(f)$
4. Naturality: The diagrams in Section 5.4 commute

5.26 Conclusion: Categorical Foundations of CEREBRUM

The category-theoretic formalization presented in this supplement provides rigorous mathematical foundations for the CEREBRUM framework. By expressing case relationships through category theory, we establish:

1. A precise language for defining model transformations
2. Provable properties of compositional operations
3. Formal verification of transformation coherence
4. Mathematical bridges between linguistics, active inference, and cognitive modeling

This formalization not only validates the theoretical consistency of CEREBRUM but also guides practical implementations by providing clear mathematical structures that should be preserved in computational systems.

Supplement 6: Future Directions - Operational Roadmap

This supplement provides a structured operational roadmap for the future development of the CEREBRUM framework, outlining actionable steps across theoretical research and practical implementation.

1. Core Framework Development

1.1 Theoretical Pathway (Conceptual Refinement & Extension)

- **Near-Term (Months)**:
  - Refine the mathematical definitions of core cases (NOM, ACC, DAT, GEN, INS, LOC, ABL, VOC) using category theory and active inference principles, clarifying precision-weighting dynamics.
  - Formalize the mathematics of the novel cases (CNJ, REC, MET, EXP, DIA, ORC, GEN), including their interaction potentials.
  - Draft initial specifications for case transformation functors, natural transformations, and associated coherence checks.
  - Begin mapping formal properties required for transformation verification (e.g., invariants, pre/post-conditions).
- **Mid-Term (Year)**:
  - Develop formal proofs for properties of case compositions and transformations (e.g., commutativity, associativity, idempotency where applicable).
  - Explore the integration of additional linguistic features (aspect, tense, modality) into the formal framework, defining their compositional semantics and interaction laws with cases.
  - Develop theoretical models for uncertainty quantification propagation during case transformations.
  - Conduct theoretical analysis of computational complexity for core and novel case transformations.
- **Long-Term (Years)**:
  - Develop a comprehensive category-theoretic model of the entire CEREBRUM ecosystem, including higher-order case structures and recursive applications.
  - Investigate deep connections and potential formal mappings to other formalisms (process calculi, type theory, control theory, quantum information theory, sheaf theory).
  - Formulate a theoretical basis for cognitive security within the case framework (Case-Based Access Control - CBAC), including information flow control properties.
  - Develop formal methods and proof strategies for verifying the correctness and safety of complex case transformation sequences.

1.2 Practical Pathway (Implementation & Tooling)

- **Near-Term (Months)**:
  - Finalize V1.0 of the language-agnostic core specification document for `CaseModel` interfaces, transformation methods, and metadata standards.
  - Implement the reference library (e.g., Python) covering core cases, basic transformations, and initial novel cases (e.g., [EXP], [DIA]).
  - Establish CI/CD pipelines for the reference library, including automated testing for core functionality.
  - Create basic visualization prototypes (static diagrams, simple animations) for illustrating individual case states and transformations.
  - Develop initial template projects or starter kits for users.
- **Mid-Term (Year)**:

  - Implement efficient algorithms for case transformations, addressing performance optimization through profiling and algorithmic improvements.
  - Develop V1.0 interactive visualization tools (e.g., web-based) for mapping transformation dynamics, case relationship networks, and precision shifts.
  - Implement robust multiple dispatch mechanisms (e.g., pattern matching, interface-based) in the reference library with clear API design.
  - Design and prototype database schemas (e.g., graph-based, document-based) with indexing strategies for storing and querying case-bearing models.
  - Establish public open-source repository with clear contribution guidelines and issue tracking.
- **Long-Term (Years)**:
  - Develop mature programming libraries in multiple key languages (functional, OO, low-level) with well-defined cross-language compatibility layers and FFI strategies.
  - Create advanced visualization suites for hierarchical ecosystem views, interactive workflow analysis, temporal dynamics, and potentially VR/AR exploration.
  - Implement specialized database solutions with optimized query languages and potentially custom storage engines for case operations at scale.
  - Develop comprehensive benchmarking tools, standard test suites, and performance profiling utilities.
  - Build initial cognitive security tools based on CBAC principles (e.g., transformation auditing logs, policy definition interfaces).
  - Explore and prototype hardware acceleration techniques (GPU, TPU, FPGA) for computationally intensive case transformations.

2. Ecosystem & Community Building

2.1 Theoretical Pathway (Community Standards, Validation & Ethics)

- **Near-Term (Months)**:
  - Draft community standards (e.g., via RFC process) for documenting case definitions, transformation properties, and formal proofs.
  - Identify key theoretical benchmarks and challenge problems for framework validation (e.g., modeling specific cognitive biases, canonical intelligence analysis scenarios).
  - Initiate discussions on ethical considerations and potential biases related to case definitions and applications.
- **Mid-Term (Year)**:
  - Establish peer-review processes within the community for theoretical extensions and contributions.
  - Define formal validation protocols and metrics for comparing CEREBRUM models against cognitive science data and task performance benchmarks.
  - Develop initial ethical guidelines for responsible development and deployment of CEREBRUM-based systems.
- **Long-Term (Years)**:
  - Curate a shared, versioned library of validated case patterns, transformation sequences, and theoretical results.
  - Foster theoretical debate and refinement through dedicated workshops, special journal issues, and online forums.
  - Establish mechanisms for ongoing review and updating of ethical guidelines based on framework evolution and application experience.

2.2 Practical Pathway (Governance, Outreach, Education & Support)

- **Near-Term (Months)**:

  - Establish initial open-source governance structure (e.g., interim steering committee) via the Active Inference Institute, defining roles and responsibilities.
  - Create foundational documentation (tutorials, API references, conceptual guides) for the reference library.
  - Launch a project website with clear communication channels (e.g., mailing list, chat server, forum).
- **Mid-Term (Year)**:
  - Formalize governance with a Technical Steering Committee (TSC) and chartered working groups (e.g., Library Dev, Theory, Applications, Documentation).
  - Develop comprehensive educational materials (interactive tutorials, course modules, video lectures, detailed case studies) explaining the framework and its usage.
  - Organize regular community calls, online hackathons, and potentially in-person workshops or sprints.
  - Establish dedicated user support channels and processes.
- **Long-Term (Years)**:
  - Implement mentorship programs to onboard and support new contributors.
  - Foster adoption in academic and industry settings through targeted outreach, demonstrations, and partnerships.
  - Establish long-term maintenance, versioning (e.g., semantic versioning), and deprecation strategies for libraries and tools.
  - Develop certification programs or standards for CEREBRUM practitioners or compliant tools.
  - Track adoption metrics, gather user feedback systematically, and publish impact case studies.

3. Interdisciplinary Integration & Application

3.1 Theoretical Pathway (Cross-Disciplinary Formalization & Modeling)

- **Near-Term (Months)**:
  - Map core concepts from related fields (e.g., affordances in ecological psychology, effect systems in PL theory, schema theory) to CEREBRUM cases and transformations.
  - Identify specific intelligence production workflows (e.g., hypothesis generation, evidence integration) suitable for initial formalization using CEREBRUM.
  - Analyze potential applications in modeling social interaction dynamics and organizational structures.
- **Mid-Term (Year)**:
  - Develop formal translations and interoperability specifications between CEREBRUM and other modeling frameworks (e.g., ACT-R modules, BPMN, process calculi structures).
  - Formalize case-based representations for specific AI tasks (e.g., LLM reasoning steps, multi-agent communication protocols, reinforcement learning state/action spaces).
  - Explore theoretical integration with AI safety frameworks (e.g., modeling value alignment constraints, specifying safe operational modes using cases).
- **Long-Term (Years)**:
  - Create unified theoretical frameworks integrating CEREBRUM with complementary approaches (e.g., CEREBRUM + session types for communication, CEREBRUM + formal verification methods).
  - Develop theoretical models for large-scale socio-technical systems, cognitive economies, or collective intelligence using case-based principles.
  - Investigate the theoretical underpinnings of emergence and self-organization in CEREBRUM model ecosystems.

3.2 Practical Pathway (Validation, Case Studies, Domain-Specific Tools & Integration)

- **Near-Term (Months)**:
  - Conduct initial proof-of-concept case studies applying CEREBRUM to simple, well-defined tasks in intelligence analysis, cognitive modeling, or system design.
  - Identify and document potential integration points and API requirements for existing AI platforms (e.g., LLM APIs, robotics middleware, simulation environments).
  - Define initial standardized data formats for representing case models and transformation histories for interchange.
- **Mid-Term (Year)**:
  - Develop domain-specific CEREBRUM extensions and libraries (e.g., toolkits for cybersecurity threat analysis, clinical pathway modeling, educational assessment, scientific discovery workflows).
  - Implement CEREBRUM-based components within larger AI systems (e.g., case-aware LLM prompters, case-based MAS coordination modules, adaptive UI components).
  - Validate CEREBRUM models against empirical data from cognitive science experiments or human performance in relevant tasks.
  - Refine and standardize data formats for model exchange and interoperability.
- **Long-Term (Years)**:
  - Build and deploy end-to-end applications leveraging CEREBRUM for complex, real-world tasks (e.g., adaptive training systems, resilient intelligence analysis platforms, personalized medicine decision support).
  - Develop standardized benchmarks, shared datasets, and evaluation methodologies for CEREBRUM applications in specific domains.
  - Demonstrate measurable improvements in efficiency, robustness, interpretability, or adaptability in practical applications compared to non-case-based approaches through rigorous evaluation.
  - Foster a marketplace or repository for pre-built CEREBRUM components and domain-specific solutions.

4. Conclusion: An Operational Vision

This operational roadmap transforms the future directions into a structured plan with distinct theoretical and practical tracks. By pursuing these parallel yet interconnected pathways across core development, community building, and interdisciplinary integration, the CEREBRUM framework can evolve from a promising concept into a robust, well-supported, and widely applicable ecosystem for advancing cognitive modeling and intelligent system design.

Supplement 7: Computational Complexity of Case Transformations

7.1 Introduction: Resource Scaling in Case-Based Cognitive Systems

The computational requirements of generative models in CEREBRUM vary significantly based on their case declensions. Each case imposes distinct resource constraints, optimization patterns, and scaling relationships with problem complexity. This supplement provides a comprehensive analysis of the computational complexity characteristics across different case assignments, with particular focus on Partially Observable Markov Decision Process (POMDP) formulations under the Free Energy Principle. We examine both theoretical bounds and practical implementations to demonstrate how intelligent resource allocation strategies can optimize overall system performance through appropriate case assignments.

7.2 Active Inference Framework for Case-Based Computational Analysis

To formalize our analysis, we adapt the traditional POMDP framework to an Active Inference perspective, defined by the tuple (S, A, T, Ω, O, F) where: - S is a finite set of states s - A is a finite set of actions a - $T : S \times A \times S \to [0, 1]$ is the transition function, where $T(s'|s, a)$ represents the probability of transitioning to state s' from state s given action a - Ω is a finite set of observations o - $O : S \times A \times \Omega \to [0, 1]$ is the observation function - F is the variational free energy, defined as $F = D_{KL}[q(s|T(m))||p(s|m)] - \mathbb{E}_p[\log p(o|s, T(m))]$

Unlike traditional POMDP formulations that incorporate reward functions, our Active Inference framework operates directly on probability distributions, using surprise minimization bounded by: 1. Variational Free Energy (F) for perception and state estimation 2. Expected Free Energy $(\mathbb{E}[\Delta F])$ for action selection and planning

Within this framework, we analyze how different case assignments affect computational resource requirements based on: 1. State space complexity 2. Belief update operations via free energy minimization 3. Policy computation complexity via expected free energy minimization 4. Precision-weighted attention allocation $\beta(c, m)$ 5. Memory requirements for historical data 6. Communication overhead between models

7.3 Computational Complexity by Case Declension

7.3.1 Nominative Case [NOM]

The nominative case, as the agent-role assignment, bears the highest computational burden for prediction generation and action selection.

State Space Considerations: - Maintains full internal state representation s - Requires access to complete model parameters θ - Active inference complexity scales with $O(|S|^2 \times |A|)$ for full policy computation via expected free energy minimization

Resource Scaling Properties: - Computational demand increases quadratically with state space size $|S|$ - Working memory requirements scale linearly with belief state dimensionality - Most sensitive to stochasticity in environment dynamics $T(s'|s, a)$

Optimization Profile: - Benefits most from predictive processing optimizations - Precomputation of policies via expected free energy minimization provides significant efficiency gains - Amortized inference approaches particularly beneficial for minimizing F

7.3.2 Accusative Case [ACC]

The accusative case, serving as the object of transformation, experiences different computational demands focused on parameter updates.

State Space Considerations: - Constrained to gradients and parameter update operations on θ - Complexity dominated by backpropagation requirements - Scales with $O(|S| \times |\theta|)$ where $|\theta|$ is the parameter space size

Resource Scaling Properties: - Computational intensity peaks during learning phases - Memory requirements increase linearly with parameter count $|\theta|$ - Optimization overhead scales with the complexity of free energy landscapes

Optimization Profile: - Benefits from sparse update mechanisms - Leverages efficient gradient calculation methods for $\frac{\partial F}{\partial \theta}$ - Focused attention on specific parameter subspaces reduces resource needs

7.3.3 Dative Case [DAT]

The dative case, as receiver of information, presents unique computational requirements centered on input processing.

State Space Considerations: - Focused on efficient sensory processing of observations o - Complexity scales with $O(|\Omega| \times |S|)$ for sensory mapping - Input filtering operations dominate computational load

Resource Scaling Properties: - Memory requirements scale with input buffer size for observations o - Processing demand correlates with input dimensionality and rate - Computational intensity concentrated at sensory interfaces

Optimization Profile: - Benefits from attention mechanisms to filter relevant inputs - Efficient encoding strategies significantly reduce complexity - Preprocessing pipelines provide substantial computational savings

7.3.4 Genitive Case [GEN]

The genitive case, functioning as a product generator, presents high asymmetric computational costs during output production.

State Space Considerations: - Maintains generative pathways for complex output synthesis - Computational complexity scales with $O(|S| \times |O_d|)$ where $|O_d|$ is output dimensionality - Resource demands vary with fidelity requirements

Resource Scaling Properties: - Computational demand increases substantially with output complexity - Memory requirements scale with output buffer size and history length - Processing intensity proportional to required output quality

Optimization Profile: - Benefits from caching intermediate generation results - Progressive generation strategies can reduce peak resource demands - Quality-resource tradeoffs offer significant optimization opportunities

7.3.5 Instrumental Case [INS]

The instrumental case, serving as a computational tool, demonstrates focused resource allocation to specific algorithmic processes.

State Space Considerations: - Maintains procedural knowledge representations - Complexity scales with $O(|A| \times |E|)$ where $|E|$ represents execution steps - Process-specific optimizations dominate efficiency gains

Resource Scaling Properties: - Computational intensity focused on algorithm execution - Memory requirements proportional to procedure complexity - Resource demands vary with procedural optimization level

Optimization Profile: - Benefits from procedure-specific hardware acceleration - Algorithm selection critically impacts resource efficiency - Just-in-time compilation provides substantial benefits

7.3.6 Locative Case [LOC]

The locative case, providing contextual environment, demonstrates distinct resource patterns related to context maintenance.

State Space Considerations: - Maintains environmental and contextual representations - Complexity scales with $O(|C| \times |I|)$ where $|C|$ is context variables and $|I|$ is interactions - Context switching operations dominate computational costs

Resource Scaling Properties: - Memory requirements increase with contextual complexity - Processing demands scale with context update frequency - Storage complexity proportional to environmental detail level

Optimization Profile: - Benefits from hierarchical context representations - Lazy context loading significantly reduces memory demands - Context caching provides substantial performance benefits

7.3.7 Ablative Case [ABL]

The ablative case, serving as historical information source, demonstrates memory-intensive computational patterns.

State Space Considerations: - Maintains historical state trajectories $s_{t-1}, s_{t-2}, ..., s_{t-h}$ and causal models - Complexity scales with $O(|H| \times |S|)$ where $|H|$ is historical depth - Temporal indexing operations dominate computational costs

Resource Scaling Properties: - Storage requirements scale linearly with historical depth $|H|$ - Processing demands increase with causal inference complexity - Memory access patterns critically impact performance

Optimization Profile: - Benefits from progressive fidelity reduction for older states - Temporal compression strategies provide significant storage savings - Selective retention policies balance resource use with information preservation

7.3.8 Vocative Case [VOC]

The vocative case, serving as an addressable interface, demonstrates unique invocation-based resource patterns.

State Space Considerations: - Maintains minimal persistent state during idle periods - Activation complexity typically constant time $O(1)$ for name recognition - Resource demands spike during activation transitions

Resource Scaling Properties: - Baseline computational requirements lowest of all cases when idle - Memory footprint minimal during dormant periods - Activation spikes create momentary high resource demands

Optimization Profile: - Benefits from hibernation strategies during inactive periods - Two-phase activation reduces false positive resource waste - Load prioritization during activation transition improves responsiveness

7.4 Comparative Resource Scaling Analysis

Table 1: Computational Complexity Analysis by Case in Active Inference Framework

Case	Time Complexity	Space Complexity	Primary Resource Bottleneck	Optimization Priority
[NOM]	$O(\|S\|^2 \times \|A\|)$	$O(\|S\| + \|A\|)$	Expected free energy minimization	Amortized inference
[ACC]	$O(\|S\| \times \|\theta\|)$	$O(\|\theta\|)$	Gradient calculation $\frac{\partial F}{\partial \theta}$	Sparse updates
[DAT]	$O(\|\Omega\| \times \|S\|)$	$O(\|\Omega\|)$	Input processing o	Attention mechanisms
[GEN]	$O(\|S\| \times \|O_d\|)$	$O(\|O_d\|)$	Output generation	Progressive generation
[INS]	$O(\|A\| \times \|E\|)$	$O(\|E\|)$	Algorithm execution	Hardware acceleration
[LOC]	$O(\|C\| \times \|I\|)$	$O(\|C\|)$	Context maintenance	Hierarchical representation
[ABL]	$O(\|H\| \times \|S\|)$	$O(\|H\| \times \|S\|)$	Historical storage	Temporal compression
[VOC]	$O(1)$ - $O(\|S\|)$	$O(1)$ - $O(\|S\|)$	Activation transition	Hibernation strategies

Where: - $|S|$ = State space size - $|A|$ = Action space size - $|\theta|$ = Parameter space size - $|\Omega|$ = Observation space size - $|O_d|$ = Output dimensionality - $|E|$ = Execution steps - $|C|$ = Context variables - $|I|$ = Interaction variables - $|H|$ = Historical depth

7.5 Precision-Weighted Resource Allocation in Active Inference

Within the active inference formulation, CEREBRUM optimizes computational resource allocation through precision-weighting mechanisms $\beta(c, m)$ that dynamically adjust resource distribution based on expected information gain. This approach leads to several important observations regarding case-based resource scaling:

1. **Precision-Driven Priority Shifting**: Resources are allocated preferentially to high-precision components of the generative model, with precision distributions varying by case assignment:
    - [NOM] cases receive maximum precision for likelihood mapping $p(o|s, \theta)$
    - [ACC] cases prioritize precision for parameter updates $\frac{\partial F}{\partial \theta}$
    - [DAT] cases emphasize precision for input processing of observations o
    - [GEN] cases maximize precision for output generation
2. **Free Energy Budgeting**: Overall system resources are allocated to minimize expected free energy $\mathbb{E}[\Delta F]$ across case-bearing components, leading to resource conservation where precision is lower.
3. **Hierarchical Memory Access**: Cases implement different memory access patterns with hierarchical precision weighting $\beta(c, m)$ determining depth and breadth of working memory allocation.

7.6 Resource Optimization Strategies for Case Transitions

CEREBRUM implementations can leverage several strategies to optimize resource utilization during case transformations $T(m)$:

1. **Just-in-Time Compilation**: Selectively compile and execute only the necessary components for the current case assignment
2. **Case-Specific Memory Management**: Implement memory allocation strategies tailored to each case's access patterns
3. **Predictive Preloading**: Anticipate case transitions $T(s'|s, a)$ and preload resources based on transition probabilities
4. **Graduated Fidelity Control**: Adjust computational precision $\beta(c, m)$ based on case-specific sensitivity requirements
5. **Parallel Case Processing**: Distribute compatible case operations across parallel computing resources

7.7 Theoretical Bounds on Case-Based Resource Optimization

We establish several theoretical bounds on the performance gains achievable through case-based resource optimization:

Theorem 1: Nominal-Vocative Efficiency Ratio For any generative model m with state space S, the ratio of computational resources required in nominative vs. vocative case is lower-bounded by $\Omega(|S|)$.

Theorem 2: Ablative Storage Efficiency For any model with historical depth $|H|$, temporal compression strategies can reduce storage requirements from $O(|H| \times |S|)$ to $O(|H| \times \log |S|)$ while preserving causal inference capabilities.

Theorem 3: Dative-Accusative Complementarity Models alternating between dative and accusative cases can achieve Pareto-optimal resource utilization when input processing (DAT) and parameter updates (ACC) are time-multiplexed.

7.8 Case Selection as Resource Optimization Strategy

Strategic case assignment emerges as a powerful resource optimization approach in complex modeling ecosystems. When multiple models have overlapping capabilities, assigning complementary cases allows the system to optimize resource utilization while maintaining functional coverage.

7.8.1 Resource-Optimal Case Assignment Algorithm

```
Algorithm 1: Resource-Optimal Case Assignment

Input: Set of models M, set of functions F, resource constraints R
Output: Case assignments C for each model in M

1. Initialize priority queue Q based on function importance
2. For each function f in F (in priority order):
   a. Identify minimal resource requirements r_f for function f
   b. Select model m from M with best performance/resource ratio for f
   c. Assign case to m that optimizes for function f
   d. Update available resources: R = R - r_f
   e. Update model capabilities based on new case assignment
3. Optimize remaining case assignments for models without critical functions
4. Return case assignments C
```

This algorithm demonstrates how CEREBRUM systems can dynamically adjust case assignments to achieve resource-optimal configurations under varying constraints.

7.9 Practical Implications for Implementation

The computational complexity characteristics of different cases directly inform implementation strategies:

1. **Hardware Acceleration Targets**:
   - FPGAs are particularly effective for [NOM] case prediction acceleration
   - GPUs provide optimal performance for [ACC] case gradient calculations $\frac{\partial F}{\partial \theta}$
   - TPUs excel at [GEN] case output generation tasks
2. **Memory Hierarchy Utilization**:
   - [NOM] and [GEN] cases benefit most from high-bandwidth memory
   - [ABL] cases can leverage tiered storage with cold/warm/hot zones
   - [VOC] cases operate effectively from cache memory during activation
3. **Distributed Computing Patterns**:
   - [DAT] cases perform well in edge computing configurations
   - [NOM] cases benefit from centralized computing resources
   - [GEN] cases can be effectively distributed across specialized processing units
4. **Scaling Constraints**:
   - [ABL] case scaling is storage-bound in most implementations
   - [NOM] case scaling is computation-bound for complex environments
   - [VOC] case scaling is primarily latency-bound during activation

7.10 Conclusion: Computational Complexity as Design Principle

The computational complexity characteristics of different case assignments provide a principled foundation for resource-aware cognitive system design. By understanding the distinct scaling properties of each case, CEREBRUM implementations can:

1. Strategically assign cases to optimize system-wide resource utilization
2. Predict performance bottlenecks before they manifest
3. Design hardware acceleration strategies aligned with case-specific demands
4. Implement precision-weighted resource allocation mechanisms $\beta(c, m)$
5. Develop case transition protocols that minimize resource contention

This analysis demonstrates that case declension not only provides a linguistic-inspired framework for understanding model relationships but also constitutes a practical resource optimization strategy for complex cognitive systems.

7.11 References

1. Friston, K. J., Parr, T., & de Vries, B. (2017). The graphical brain: belief propagation and active inference. Network Neuroscience, 1(4), 381-414.
2. Kaelbling, L. P., Littman, M. L., & Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains. Artificial intelligence, 101(1-2), 99-134.
3. Silver, D., & Veness, J. (2010). Monte-Carlo planning in large POMDPs. Advances in neural information processing systems, 23.
4. Gershman, S. J. (2019). What does the free energy principle tell us about the brain? Neurons, Behavior, Data analysis, and Theory, 2(3), 1-10.
5. Da Costa, L., Parr, T., Sajid, N., Veselic, S., Neacsu, V., & Friston, K. (2020). Active inference on discrete state-spaces: A synthesis. Journal of Mathematical Psychology, 99, 102447.
6. Sajid, N., Ball, P. J., Parr, T., & Friston, K. J. (2021). Active inference: demystified and compared. Neural Computation, 33(3), 674-712.
7. Millidge, B., Seth, A., & Buckley, C. L. (2021). Predictive coding: a theoretical and experimental review. arXiv preprint arXiv:2107.12979.

Supplement 8: Active Inference Formulation Details

This supplement provides detailed mathematical derivations and formulations connecting the Active Inference framework to the CEREBRUM case system.

8.1 Generative Model Specification

The generative model underlying CEREBRUM's active inference framework can be specified as a tuple $(S, A, T, \Omega, O, F, \pi)$ where:

- S represents the space of hidden states, which includes both environmental states and the internal states of models in various cases.
- A represents the action space, which crucially includes case transformation operations alongside traditional actions.
- $T : S \times A \to P(S)$ specifies state transition dynamics, mapping current state and action to a probability distribution over next states.
- Ω is the space of possible observations available to models.
- $O : S \to P(\Omega)$ is the likelihood mapping from states to observations.
- F is the free energy functional that models minimize.
- π represents precision parameters modulating the influence of various probabilistic terms.

For a model M with case C, the generative model is instantiated with case-specific parameters:

$$p(o_{1:T}, s_{1:T}, \pi|C) = p(s_1|C) \prod_{t=1}^{T} p(o_t|s_t, C)p(s_t|s_{t-1}, a_{t-1}, C)p(\pi|C)$$

This factorization represents how future observations and states depend on the current case assignment, with case-specific priors, likelihood mappings, and transition dynamics.

8.2 Free Energy Principle in CEREBRUM

The Variational Free Energy (VFE) functional for a model with case C is defined as:

$$F[q, C] = D_{KL}[q(s, \pi)||p(s, \pi|o, C)] - \log p(o|C)$$

This can be reformulated as:

$$F[q, C] = E_q[\log q(s, \pi) - \log p(s, \pi, o|C)]$$

Which decomposes into:

$$F[q, C] = \underbrace{D_{KL}[q(s, \pi)||p(s, \pi|C)]}_{\text{Complexity}} - \underbrace{E_q[\log p(o|s, C)]}_{\text{Accuracy}}$$

Further expanding the complexity term:

$$D_{KL}[q(s, \pi)||p(s, \pi|C)] = \int q(s, \pi) \log \frac{q(s, \pi)}{p(s, \pi|C)} ds d\pi$$

And expanding the accuracy term:

$$E_q[\log p(o|s,C)] = \int q(s,\pi) \log p(o|s,C) ds d\pi$$

These terms have specific interpretations in CEREBRUM:

1. **Complexity** measures the divergence between the approximate posterior and the prior, which represents the computational cost of updating beliefs when adopting a case.
2. **Accuracy** measures how well the model with a given case explains observations, which represents the explanatory power of a case assignment.

For a time series of observations $o_{1:T}$, the path integral of free energy is:

$$\mathcal{F}[q, C, o_{1:T}] = \sum_{t=1}^{T} F[q_t, C]$$

Where q_t represents the evolving belief distribution at time t. The principle of least action dictates that natural systems minimize this path integral.

8.2.1 Relationship Between Free Energy and Case Transformations

Case transformations in CEREBRUM can be characterized by changes in the free energy landscape:

$$\Delta F[C_1 \to C_2] = F[q, C_2] - F[q, C_1]$$

Successful case transformations decrease the total free energy of the system. For a composite system with multiple models in different cases, the total free energy is:

$$F_{total} = \sum_i F[q_i, C_i] + \sum_{i,j} I[q_i, q_j | C_i, C_j]$$

Where $I[q_i, q_j|C_i, C_j]$ represents the mutual information between models in different cases.

8.2.2 Mapping Between Active Inference and CEREBRUM

Active Inference Concept	CEREBRUM Implementation	Mathematical Formulation
Generative Model	Case-Parametrized Model	$p(o, s, \pi\|C)$
Variational Density	Case-Specific Beliefs	$q(s, \pi\|C)$
Free Energy	Case-Specific Free Energy	$F[q, C]$
Policy Evaluation	Case Transformation Selection	$G(\pi_{C_1 \to C_2})$
Precision Parameters	Case-Modulated Parameters	π_C
Belief Updating	Case-Specific Message Passing	$q(s_i) \propto \exp(E_{q(\setminus s_i)}[\log p(o, s, \pi\|C)])$

Active Inference Concept	CEREBRUM Implementation	Mathematical Formulation
Expected Free Energy	Case Transformation Planning	$G(\pi) = E_{q(o,s\|\pi)}[\log q(s\|\pi) - \log p(o, s\|\pi)]$
Markov Blankets	Case Boundaries	$\text{Case}(M) \subseteq \text{MB}(M)$
Hierarchical Models	Case Transformation Sequences	$C_1 \to C_2 \to C_3$
Action Selection	Case-Based Policy	$\pi^*(C) = \arg\min_\pi G(\pi\|C)$
Prediction Errors	Case-Specific Prediction Errors	$\varepsilon_C = o - g(s, C)$
Active Inference	Active Case Management	Minimizing $F[q, C]$ through case selection

8.2.3 Case-Specific Free Energy Applications

Different cases in CEREBRUM implement specialized forms of free energy minimization, aligning with their functional roles in the system:

$$F_{NOM}[q] = D_{KL}[q(s)||p(s|C_{NOM})] - \alpha_{NOM} \cdot E_q[\log p(o|s, C_{NOM})]$$

$$F_{ACC}[q] = \beta_{ACC} \cdot D_{KL}[q(s)||p(s|C_{ACC})] - E_q[\log p(o|s, C_{ACC})]$$

$$F_{DAT}[q] = D_{KL}[q(s)||p(s_1, s_2|C_{DAT})] - E_q[\log p(o|s, C_{DAT})]$$

$$F_{INS}[q] = D_{KL}[q(s, a)||p(s, a|C_{INS})] - E_q[\log p(o|s, a, C_{INS})]$$

$$F_{GEN}[q] = D_{KL}[q(s)||p(s|C_{GEN})] - E_q[\log p(o|s, C_{GEN})] + \lambda_{GEN} \cdot E_q[\log p(r|s, C_{GEN})]$$

Where: - α_{NOM} represents increased precision on accuracy for Nominative case - β_{ACC} represents increased precision on complexity for Accusative case - $p(s_1, s_2|C_{DAT})$ captures the mediating role of Dative case connecting two state spaces - $p(s, a|C_{INS})$ jointly represents states and actions in Instrumental case - $\lambda_{GEN} \cdot E_q[\log p(r|s, C_{GEN})]$ adds a relational term for Genitive case

These specialized forms enable principled adaptation of free energy minimization to different functional contexts.

Information Flow Patterns Across Case Transformations The following table characterizes how information flows change during case transformations:

Source Case	Target Case	Information Flow Pattern	Free Energy Change
NOM → ACC	$F_{NOM} \to F_{ACC}$	Top-down to bottom-up	$\Delta F \propto -D_{KL}[q(o)\|p(o\|C_{ACC})]$
ACC → NOM	$F_{ACC} \to F_{NOM}$	Bottom-up to top-down	$\Delta F \propto -D_{KL}[p(s\|C_{NOM})\|q(s)]$

Source Case	Target Case	Information Flow Pattern	Free Energy Change
NOM → DAT	$F_{NOM} \rightarrow F_{DAT}$	Prediction to mediation	$\Delta F \propto -I[s_1; s_2 \| C_{DAT}]$
ACC → DAT	$F_{ACC} \rightarrow F_{DAT}$	Error correction to mediation	$\Delta F \propto -E_q[\log p(s_2 \| s_1, C_{DAT})]$
DAT → INS	$F_{DAT} \rightarrow F_{INS}$	Mediation to action-oriented	$\Delta F \propto -E_q[\log p(o \| s, a, C_{INS})]$
NOM → GEN	$F_{NOM} \rightarrow F_{GEN}$	Prediction to relation	$\Delta F \propto -\lambda_{GEN} \cdot E_q[\log p(r \| s, C_{GEN})]$
ACC → INS	$F_{ACC} \rightarrow F_{INS}$	Error-correction to action	$\Delta F \propto -E_q[\log p(a \| s, C_{INS})]$

Each transformation induces characteristic changes in the free energy landscape, with information flow redirected according to the functional role of the target case. The proportionality relations ($\Delta F \propto$) capture the dominant terms determining whether a transformation increases or decreases free energy.

Multi-Scale Free Energy Minimization CEREBRUM implements free energy minimization across multiple scales:

1. **Within-Case Scale**: Each model minimizes free energy according to its current case assignment
$$F_i[q_i, C_i] \rightarrow \min$$
2. **Transformation Scale**: Case transformations are selected to minimize expected free energy
$$G(\pi_{C_1 \rightarrow C_2}) \rightarrow \min$$
3. **System Scale**: The configuration of cases across all models minimizes total system free energy
$$F_{total} = \sum_i F[q_i, C_i] + \sum_{i,j} I[q_i, q_j | C_i, C_j] \rightarrow \min$$

This multi-scale optimization aligns with the nested Markov blanket formulation in the Free Energy Principle, where each scale offers a distinct perspective on the same underlying dynamics.

Each case modifies this formulation by emphasizing different components:

- **Nominative Case [NOM]**: Emphasizes accuracy of predictions, with higher precision on the likelihood term.
- **Accusative Case [ACC]**: Emphasizes complexity reduction through effective belief updates.
- **Dative Case [DAT]**: Balances complexity and accuracy for information mediation.
- **Instrumental Case [INS]**: Emphasizes action-dependent state transitions.

For case transformations, we define a transformation-specific free energy:

$$F_{\text{trans}}[q, C_{\text{source}} \rightarrow C_{\text{target}}] = F[q, C_{\text{target}}] + D_{KL}[q_{\text{target}}(s, \pi) || q_{\text{source}}(s, \pi)]$$

The additional KL divergence term represents the transformation cost between source and target case parametrizations.

8.3 Message Passing Schemes

For a model M with case C, belief updates follow a variational message passing scheme. Assuming factorized approximate posteriors:

$$q(s, \pi) = \prod_i q(s_i) \prod_j q(\pi_j)$$

The update equations for each factor take the form:

$$q(s_i) \propto \exp(E_{q(\setminus s_i)}[\log p(o, s, \pi|C)])$$

$$q(\pi_j) \propto \exp(E_{q(\setminus \pi_j)}[\log p(o, s, \pi|C)])$$

where $q(\setminus x)$ denotes the approximate posterior for all variables except x.

These updates are implemented as message passing operations, with the form of messages determined by the model's case. For example:

- **Nominative [NOM]** messages emphasize prediction generation:

$$m_{\text{NOM}}(s_i \to o) = E_{q(s_{\setminus i})}[\log p(o|s, C = \text{NOM})]$$

- **Accusative [ACC]** messages emphasize belief updates:

$$m_{\text{ACC}}(o \to s_i) = E_{q(s_{\setminus i})}[\log p(o|s, C = \text{ACC})]$$

- **Dative [DAT]** messages emphasize mediation:

$$m_{\text{DAT}}(s_i \to s_j) = E_{q(s_{\setminus \{i,j\}})}[\log p(s_j|s_i, C = \text{DAT})]$$

8.4 Expected Free Energy (EFE) for Case Selection

Case selection in CEREBRUM follows an active inference approach, where the Expected Free Energy (EFE) of different case transformation policies is evaluated. For a policy π that includes transforming to case C_{target}, the EFE is:

$$G(\pi) = E_{q(o,s|\pi)}[\log q(s|\pi) - \log p(o, s|\pi)]$$

This can be decomposed into:

$$G(\pi) = \underbrace{E_{q(s|\pi)}[D_{KL}[q(o|s, \pi)||p(o|s)]]}_{\text{Epistemic Value (Exploration)}} + \underbrace{E_{q(o,s|\pi)}[\log q(o|s, \pi) - \log p(o)]}_{\text{Pragmatic Value (Exploitation)}}$$

For case transformation policies, this becomes:

$$G(\pi_{C_{\text{source}} \to C_{\text{target}}}) = \underbrace{E_{q(s|C_{\text{target}})}[D_{KL}[q(o|s, C_{\text{target}})||p(o|s)]]}_{\text{Information Gain from New Case}} + \underbrace{E_{q(o,s|C_{\text{target}})}[\log q(o|s, C_{\text{target}}) - \log p(o|C_{\text{preferre}}}_{\text{Goal Alignment of New Case}}$$

The optimal case transformation policy minimizes this expected free energy:

$$\pi^* = \arg\min_{\pi} G(\pi)$$

In practice, a softmax function converts these EFE values into a probability distribution over policies:

$$p(\pi|o) \propto \exp(-\gamma \cdot G(\pi))$$

where γ is an inverse temperature parameter controlling the randomness of policy selection.

8.5 Precision Dynamics

Precision parameters in CEREBRUM modulate the influence of different probabilistic terms, affecting both inference within a case and case selection dynamics. For a model with case C, precision parameters π are updated according to:

$$q(\pi_i) \propto p(\pi_i|C) \cdot \exp(-\frac{1}{2}\pi_i \cdot \varepsilon_i^T \varepsilon_i)$$

where ε_i represents prediction errors associated with the i-th component of the generative model.

Case-specific precision defaults establish the characteristic behavior of each case:

- **Nominative [NOM]**: Higher precision on generative parameters ($\pi_{\text{gen}} > \pi_{\text{prior}}$)
- **Accusative [ACC]**: Higher precision on updating parameters ($\pi_{\text{update}} > \pi_{\text{gen}}$)
- **Genitive [GEN]**: Higher precision on relational parameters ($\pi_{\text{rel}} > \pi_{\text{other}}$)

During case transformations, precision parameters undergo structured remapping:

$$\pi_{\text{target}} = f_{C_{\text{source}} \to C_{\text{target}}}(\pi_{\text{source}})$$

This remapping function f implements case-specific precision dynamics that control how attention and computational resources are allocated after transformation.

8.6 Connections to POMDPs

The CEREBRUM Active Inference formulation extends the traditional POMDP framework in several key ways:

1. **State Space Expansion**: States include not just environmental variables but also case assignments, interface configurations, and precision parameters.
2. **Action Space Enrichment**: Actions include case transformations alongside traditional actions, enabling models to modify their functional roles.
3. **Policy Evaluation**: Unlike standard POMDPs that maximize expected reward, CEREBRUM minimizes expected free energy, balancing exploration (information gain) and exploitation (goal achievement).
4. **Belief Dynamics**: While POMDPs update beliefs using Bayes' rule, CEREBRUM implements variational belief updates that can vary based on case assignment.

The mapping between POMDP and CEREBRUM components can be formalized as:

POMDP Component	CEREBRUM Extension
States s	States s + Case assignment C
Actions a	Actions a + Case transformations
Transition $T(s'\|s,a)$	Case-dependent transitions $T(s'\|s,a,C)$
Observations o	Observations with case-specific attention o_C
Observation model $O(o\|s)$	Case-dependent observation model $O(o\|s,C)$
Reward function $R(s,a)$	Free energy minimization $F[q,C]$
Value function $V(b)$	Expected free energy $G(\pi)$

This mapping shows how CEREBRUM specializes the POMDP framework through its case-based structure and free energy optimization approach.

8.7 Neurobiological Connections and Computational Complexity

8.7.1 Neurobiological Plausibility of Case-Based Active Inference

CEREBRUM's case-based active inference formulation connects to several neurobiological mechanisms:

CEREBRUM Component	Neurobiological Correlate	Functional Role
Case Assignment	Neuromodulation	Context-dependent processing modes
Precision Parameters	Dopaminergic/Cholinergic Modulation	Attentional allocation and learning rate control
Message Passing	Canonical Microcircuits	Implementation of predictive coding
Case Transformation	Neural Gain Control	Dynamic reconfiguration of functional connectivity
Free Energy Minimization	Hierarchical Predictive Processing	Prediction error minimization across cortical hierarchies
Expected Free Energy	Prefrontal Planning	Counterfactual reasoning about future states
Markov Blankets	Functional Segregation	Maintaining conditional independence between neural subsystems

The mapping demonstrates how CEREBRUM's formal apparatus aligns with empirical findings in neuroscience, particularly regarding:

1. **Multiple Simultaneous Objectives**: The brain optimizes multiple objectives simultaneously (accuracy, complexity, exploration), which maps to CEREBRUM's case-specific free energy formulations.
2. **Context-Sensitivity**: Neural circuits reconfigure based on contextual demands, similar to case transformations in CEREBRUM.

3. **Hierarchical Processing**: The brain implements hierarchical predictive processing, with distinct information flows matching CEREBRUM's case-specific message passing schemes.

8.7.2 Computational Complexity Implications

The computational complexity of active inference in CEREBRUM varies by case:

Case	Time Complexity	Space Complexity	Dominant Operation
NOM	$O(n)$	$O(n)$	Forward prediction
ACC	$O(n \log n)$	$O(n)$	Belief update
DAT	$O(nm)$	$O(n+m)$	Information mediation
GEN	$O(n^2)$	$O(n^2)$	Relational modeling
INS	$O(na)$	$O(n+a)$	Action selection
VOC	$O(\log n)$	$O(1)$	Attention allocation

Where: - n is the dimensionality of state space - m is the dimensionality of connected model's state space - a is the dimensionality of action space

This complexity analysis reveals an important tradeoff: case assignments effectively manage computational resources by directing attention to specific aspects of inference. The system can strategically transform between cases to optimize computational efficiency based on current demands.

For a system with k models in potentially different cases, the total computational complexity is bounded by:

$$O\left(\sum_{i=1}^{k} C(C_i) + \sum_{i,j} T(C_i, C_j)\right)$$

Where $C(C_i)$ is the complexity of inference in case C_i and $T(C_i, C_j)$ is the transformation cost between cases.

This formulation demonstrates how CEREBRUM achieves scalable active inference through distributed processing and strategic case management, enabling computationally efficient implementation of the Free Energy Principle in complex systems.

8.8 Comparison with Other Active Inference Frameworks

CEREBRUM's case-based formulation of active inference extends traditional frameworks in several key directions. This section provides a mathematical comparison with other prominent active inference formalisms.

8.8.1 Comparative Free Energy Formulations

Framework	Free Energy Formulation	Key Distinguishing Features
Standard Active Inference	$F[q] = D_{KL}[q(s) \| p(s)] - E_q[\log p(o \| s)]$	Single-model inference with fixed functional role

Framework	Free Energy Formulation	Key Distinguishing Features
Hierarchical Active Inference	$F[q] = \sum_i D_{KL}[q(s_i)\|p(s_i\|s_{i+1})] - E_q[\log p(o\|s_1)]$	Fixed hierarchical message passing structure
Deep Active Inference	$F[q] = D_{KL}[q(s,\theta)\|p(s,\theta)] - E_q[\log p(o\|s,\theta)]$	Parameterizes generative model with neural networks
CEREBRUM	$F[q,C] = D_{KL}[q(s,\pi\|C)\|p(s,\pi\|C)] - E_q[\log p(o\|s,C)]$	Case-parameterized inference with dynamic functional roles

The key mathematical distinction of CEREBRUM is the explicit parameterization of both the generative model and approximate posterior by the case assignment C, which enables dynamic reconfiguration of functional roles through case transformations.

8.8.2 Relationship to Variational Message Passing

Standard variational message passing updates take the form:

$$\log q(s_i) = E_{q(s_{\setminus i})}[\log p(o,s)] + \text{const}$$

CEREBRUM generalizes this to case-dependent message passing:

$$\log q(s_i|C) = E_{q(s_{\setminus i}|C)}[\log p(o,s|C)] + \text{const}$$

The case parameter C modifies both the form of the joint distribution $p(o,s|C)$ and the factorization of the approximate posterior $q(s|C)$.

8.8.3 Extensions to Expected Free Energy

Standard expected free energy is formulated as:

$$G(\pi) = E_{q(o,s|\pi)}[\log q(s|\pi) - \log p(o,s|\pi)]$$

CEREBRUM extends this to include case transformations in the policy space:

$$G(\pi_{C_1 \to C_2}) = E_{q(o,s|C_2)}[\log q(s|C_2) - \log p(o,s|C_2)] + \gamma \cdot D_{KL}[q(s|C_2)\|q(s|C_1)]$$

Where the additional term $\gamma \cdot D_{KL}[q(s|C_2)\|q(s|C_1)]$ represents the transformation cost weighted by precision parameter γ.

8.8.4 Mathematical Advances Over Prior Work

CEREBRUM makes several mathematical contributions to active inference theory:

1. **Case-Parameterized Generative Models**: The explicit conditioning of the generative model on case assignment $p(o,s|C)$ provides a formal mechanism for dynamic reconfiguration of functional roles.

2. **Transformation-Specific Free Energy**: The introduction of transformation costs in the free energy functional F_{trans} enables principled evaluation of case transformations.
3. **Multi-Scale Free Energy Minimization**: The hierarchical organization of free energy minimization (within-case, transformation, system) provides a formal account of nested optimization processes.
4. **Precision-Weighted Case Selection**: The softmax formulation of case selection with precision weighting $\beta(c, m)$ formalizes how systems adaptively allocate resources across competing functional roles.

These advances extend active inference beyond its traditional formulation as a theory of brain function to a more general computational framework for adaptive systems with dynamic functional reconfiguration.

Supplement 9: Mathematical Foundations

This supplement provides the mathematical foundations underlying the CEREBRUM framework, detailing the formal definitions, theorems, and proofs that support the case-based cognitive architecture.

9.1 Category Theory Foundations

9.1.1 Category of Cases

We begin by formalizing the notion of linguistic cases as a mathematical category.

Definition 9.1.1 (Category of Cases). The category $\mathcal{C}$ of CEREBRUM cases consists of: - **Objects**: Linguistic cases (e.g., Nominative, Accusative, Genitive, etc.) - **Morphisms**: Case transformations $f : C_1 \to C_2$ between cases - **Composition**: Sequential application of transformations $g \circ f : C_1 \to C_3$ for $f : C_1 \to C_2$ and $g : C_2 \to C_3$ - **Identity**: The identity transformation $\text{id}_C : C \to C$ for each case C

Theorem 9.1.1 (Well-formed Category). The category $\mathcal{C}$ of cases is well-formed, satisfying: 1. Associativity: $(h \circ g) \circ f = h \circ (g \circ f)$ for all composable transformations 2. Identity: $f \circ \text{id}_C = f$ and $\text{id}_D \circ f = f$ for any transformation $f : C \to D$

Proof: The associativity follows from the sequential nature of transformations, as they represent adjustments to model interfaces and precision weights. The identity properties follow from the definition of the identity transformation as one that preserves all interface configurations and precision parameters. □

9.1.2 Functors and Natural Transformations

Case-bearing models can be viewed through the lens of functors between categories.

Definition 9.1.2 (Model Functor). A case-bearing model M defines a functor $F_M : \mathcal{C} \to \mathcal{S}$ where $\mathcal{C}$ is the category of cases and $\mathcal{S}$ is the category of model states with: - For each case C, $F_M(C)$ is the state space of model M in case C - For each transformation $f : C_1 \to C_2$, $F_M(f) : F_M(C_1) \to F_M(C_2)$ is the corresponding state transformation

Theorem 9.1.2 (Natural Transformation of Models). Given two models M and N, a coherent mapping between them across all cases forms a natural transformation $\eta : F_M \Rightarrow F_N$ between functors.

Proof: For any case transformation $f : C_1 \to C_2$, the naturality square commutes: $F_N(f) \circ \eta_{C_1} = \eta_{C_2} \circ F_M(f)$, due to the consistent relationship between models across different cases. □

9.2 Free Energy and Active Inference

9.2.1 Free Energy Principle

The free energy principle provides the theoretical basis for CEREBRUM's inference mechanisms.

Definition 9.2.1 (Variational Free Energy). For a model M with internal state s and observations o, the variational free energy is defined as:

$$F(s, o) = D_{KL}[q(h|s)||p(h|o)] - \ln p(o)$$

where $q(h|s)$ is the recognition density over hidden states h implied by internal state s, $p(h|o)$ is the true posterior, and $p(o)$ is the evidence.

Theorem 9.2.1 (Free Energy Decomposition). The variational free energy can be decomposed into:

$$F(s,o) = \underbrace{E_{q(h|s)}[-\ln p(o|h)]}_{\text{Accuracy}} + \underbrace{D_{KL}[q(h|s)||p(h)]}_{\text{Complexity}}$$

Proof: Starting with Definition 9.2.1, we apply Bayes' rule to the posterior:

$$\begin{aligned}
F(s,o) &= D_{KL}[q(h|s)||p(h|o)] - \ln p(o) \\
&= \int q(h|s) \ln \frac{q(h|s)}{p(h|o)} dh - \ln p(o) \\
&= \int q(h|s) \ln \frac{q(h|s)}{p(h) \cdot p(o|h)/p(o)} dh - \ln p(o) \\
&= \int q(h|s) \ln \frac{q(h|s) \cdot p(o)}{p(h) \cdot p(o|h)} dh - \ln p(o)
\end{aligned}$$

Distributing the logarithm and separating terms:

$$\begin{aligned}
F(s,o) &= \int q(h|s) \ln \frac{q(h|s)}{p(h)} dh + \int q(h|s) \ln \frac{p(o)}{p(o|h)} dh - \ln p(o) \\
&= D_{KL}[q(h|s)||p(h)] + \int q(h|s) \ln p(o) dh - \int q(h|s) \ln p(o|h) dh - \ln p(o)
\end{aligned}$$

Since $\int q(h|s) dh = 1$, we have $\int q(h|s) \ln p(o) dh = \ln p(o)$, giving:

$$F(s,o) = D_{KL}[q(h|s)||p(h)] - \int q(h|s) \ln p(o|h) dh$$

The second term is the negative expected log likelihood, which measures the (in)accuracy of predictions. □

9.2.2 Case-Specific Free Energy

In CEREBRUM, free energy is case-dependent due to differences in precision weighting.

Definition 9.2.2 (Case-Specific Free Energy). For a model M in case C with precision parameters π_C, the case-specific free energy is:

$$F_C(s,o) = E_{q(h|s)}[-\ln p(o|h;\pi_C)] + D_{KL}[q(h|s)||p(h)]$$

where $p(o|h;\pi_C)$ is the likelihood function with case-specific precision weighting.

Theorem 9.2.2 (Free Energy Transformation). For cases C_1 and C_2 with a transformation $f : C_1 \to C_2$, the free energies relate as:

$$F_{C_2}(f(s),o) = F_{C_1}(s,o) + \Delta F_f(s,o)$$

where $\Delta F_f(s, o)$ is the transformation impact on free energy.

Proof: The transformation f modifies both the internal state representation and the precision parameters. Let π_{C_1} and π_{C_2} be the precision parameters for cases C_1 and C_2 respectively. Then:

$$F_{C_2}(f(s), o) = E_{q(h|f(s))}[-\ln p(o|h; \pi_{C_2})] + D_{KL}[q(h|f(s))||p(h)]$$
$$F_{C_1}(s, o) = E_{q(h|s)}[-\ln p(o|h; \pi_{C_1})] + D_{KL}[q(h|s)||p(h)]$$

The difference $\Delta F_f(s, o)$ arises from: 1. Changes in the recognition density: $q(h|f(s))$ vs. $q(h|s)$ 2. Changes in precision weighting: π_{C_2} vs. π_{C_1}

These contribute to the overall transformation impact. □

9.3 Precision and Uncertainty

9.3.1 Precision Matrices and Uncertainty

Precision plays a central role in CEREBRUM's case representations.

Definition 9.3.1 (Precision Matrix). For a Gaussian likelihood model $p(o|h) = \mathcal{N}(g(h), \Sigma)$, the precision matrix is $\Pi = \Sigma^{-1}$, where $g(h)$ is the generative mapping from hidden states to observations.

Theorem 9.3.1 (Precision and Free Energy). Increased precision on a specific observation dimension reduces free energy when predictions are accurate, but increases it when predictions are inaccurate.

Proof: For a univariate Gaussian likelihood with precision π, the contribution to free energy is:

$$-\ln p(o|h) = \frac{1}{2}\ln(2\pi/\pi) + \frac{\pi}{2}(o - g(h))^2$$

Taking the derivative with respect to precision:

$$\frac{\partial(-\ln p(o|h))}{\partial \pi} = -\frac{1}{2\pi} + \frac{1}{2}(o - g(h))^2$$

This is negative when $(o - g(h))^2 < 1/\pi$, meaning prediction errors are small relative to expected variance, and positive otherwise. □

9.3.2 Case-Specific Precision Allocation

Different cases allocate precision differently across observation dimensions.

Definition 9.3.2 (Case-Specific Precision Profile). A case C defines a precision profile $P_C : D \to \mathbb{R}^+$ mapping each dimension $d \in D$ of the observation space to a precision weight.

Theorem 9.3.2 (Optimal Case Selection). Given observations o, the optimal case C^* minimizes expected free energy:

$$C^* = \arg\min_C E_{p(o'|o)}[F_C(s, o')]$$

where $p(o'|o)$ is the predictive distribution over future observations.

Proof: This follows directly from the active inference principle of minimizing expected free energy. The case that assigns precision optimally for future observations will minimize the expected free energy. □

9.4 Multiple Dispatch and Case Polymorphism

9.4.1 Formal Definition of Multiple Dispatch

Multiple dispatch provides the mathematical basis for case-specific operations.

Definition 9.4.1 (Multiple Dispatch). A multiple dispatch function δ maps an operation ω, a case C, and arguments $\mathbf{a}$ to an implementation:

$$\delta : \Omega \times \mathcal{C} \times A \to I$$

where Ω is the set of operations, $\mathcal{C}$ is the set of cases, A is the argument space, and I is the implementation space.

Theorem 9.4.1 (Dispatch Consistency). For any operation ω, cases C_1 and C_2, and transformation $f : C_1 \to C_2$, the results of dispatch are consistent:

$$\delta(\omega, C_2, f(\mathbf{a})) \circ f = f \circ \delta(\omega, C_1, \mathbf{a})$$

where composition denotes sequential application.

Proof: This follows from the requirement that case transformations preserve the semantics of operations while adapting their implementations to case-specific contexts. □

9.4.2 Case Polymorphism

Case polymorphism generalizes object-oriented polymorphism to linguistic cases.

Definition 9.4.2 (Case Polymorphism). A case-polymorphic function ϕ has different implementations for different cases while maintaining a consistent interface:

$$\phi_C : A \to B_C$$

where A is the input type and B_C is the case-specific output type.

Theorem 9.4.2 (Polymorphic Coherence). For a case-polymorphic function ϕ and transformation $f : C_1 \to C_2$, there exists a mapping $f_B : B_{C_1} \to B_{C_2}$ such that:

$$\phi_{C_2} \circ f = f_B \circ \phi_{C_1}$$

Proof: The existence of f_B follows from the requirement that case transformations preserve the semantic relationships between inputs and outputs across different implementations. □

9.5 Information Geometry of Case Spaces

9.5.1 Case Manifold

The space of cases can be viewed as a manifold with an information-geometric structure.

Definition 9.5.1 (Case Manifold). The manifold $\mathcal{M}$ of cases is a geometric structure where: - Points represent individual cases - Tangent vectors represent infinitesimal case transformations - The metric is derived from the Kullback-Leibler divergence between recognition densities

Theorem 9.5.1 (Fisher Information Metric). The natural metric on the case manifold is the Fisher information metric:

$$g_{\mu\nu}(\theta) = E_{p(x|\theta)}\left[\frac{\partial \log p(x|\theta)}{\partial \theta^\mu}\frac{\partial \log p(x|\theta)}{\partial \theta^\nu}\right]$$

where θ parametrizes the case and $p(x|\theta)$ is the case-conditional probability.

Proof: This follows from information geometry principles, where the KL divergence between nearby distributions induces a Riemannian metric that corresponds to the Fisher information. □

9.5.2 Geodesics and Optimal Transformations

The geometry of the case manifold determines optimal transformation paths.

Definition 9.5.2 (Transformation Geodesic). A geodesic path between cases C_1 and C_2 on the manifold $\mathcal{M}$ represents the optimal transformation sequence minimizing information loss.

Theorem 9.5.2 (Minimum Free Energy Path). The geodesic path between cases C_1 and C_2 minimizes the integrated free energy change:

$$\gamma^* = \arg\min_\gamma \int_0^1 \Delta F(s, o, \gamma(t))\, dt$$

where $\gamma : [0,1] \to \mathcal{M}$ is a path with $\gamma(0) = C_1$ and $\gamma(1) = C_2$.

Proof: The proof follows from calculus of variations, showing that the Euler-Lagrange equations for this functional yield the geodesic equation in the Fisher information metric. □

9.6 Topological Data Analysis of Case Structures

9.6.1 Persistent Homology

Topological features provide insights into case structure invariants.

Definition 9.6.1 (Persistence Diagram). For a case structure filtration $\{K_\alpha\}_{\alpha \geq 0}$, the persistence diagram $\text{Dgm}_p(K)$ captures the birth and death times of p-dimensional topological features.

Theorem 9.6.1 (Case Structure Invariants). Topological invariants of case structures are preserved under continuous transformations.

Proof: This follows from the functoriality of homology and the stability theorem for persistence diagrams, which ensures that small perturbations in the case structure lead to small changes in the diagram. □

9.6.2 Mapper Algorithm for Case Visualization

The Mapper algorithm provides a topological representation of case spaces.

Definition 9.6.2 (Case Mapper). The Mapper algorithm applied to case space produces a simplicial complex representation: 1. Define a filter function $f : \mathcal{C} \to \mathbb{R}^d$ 2. Cover $\mathbb{R}^d$ with overlapping sets $\{U_i\}$ 3. For each U_i, cluster $f^{-1}(U_i) \cap \mathcal{C}$ 4. Create a vertex for each cluster and an edge for each non-empty intersection

Theorem 9.6.2 (Topological Representativeness). The Mapper complex captures essential topological features of the case space that persist across scales.

Proof: By the nerve theorem, under appropriate conditions, the Mapper complex is homotopy equivalent to the underlying space, preserving key topological features. □

9.7 Dynamical Systems Perspective

9.7.1 Vector Fields and Flows

Case models can be viewed through the lens of dynamical systems.

Definition 9.7.1 (Free Energy Gradient Flow). The dynamics of a case model follows the negative gradient of free energy:

$$\frac{ds}{dt} = -\nabla_s F_C(s, o)$$

where s is the model state and F_C is the case-specific free energy.

Theorem 9.7.1 (Fixed Points and Stability). The fixed points of the gradient flow correspond to free energy minima, with stability determined by the Hessian of free energy.

Proof: At fixed points, $\nabla_s F_C(s, o) = 0$. The stability is determined by the eigenvalues of the Hessian matrix $H = \nabla_s^2 F_C(s, o)$. If all eigenvalues are positive, the fixed point is a stable minimum. □

9.7.2 Bifurcations and Case Transitions

Case transformations can induce bifurcations in system dynamics.

Definition 9.7.2 (Case Bifurcation). A case transformation $f : C_1 \to C_2$ induces a bifurcation if the qualitative structure of fixed points changes across the transformation.

Theorem 9.7.2 (Bifurcation Conditions). A case transformation induces a bifurcation if there exists a state s such that the Hessian $\nabla_s^2 F_{C_1}(s, o)$ has a zero eigenvalue and the corresponding eigenvector is not in the kernel of the transformation's third derivative tensor.

Proof: This follows from bifurcation theory, specifically the conditions for a saddle-node bifurcation in dynamical systems. □

9.8 Computational Complexity

9.8.1 Complexity of Case Operations

The computational requirements of CEREBRUM operations are case-dependent.

Definition 9.8.1 (Operation Complexity). The computational complexity of an operation ω in case C with parameters π is denoted $\kappa(\omega, C, \pi)$.

Theorem 9.8.1 (Transformation Complexity). The complexity of a case transformation $f : C_1 \to C_2$ is bounded by:

$$\kappa(f, C_1, C_2, \pi) = O(P \cdot D)$$

where P is the number of parameters and D is the dimensionality of the interface.

Proof: The transformation requires mapping P parameters across D interface dimensions in the worst case, giving the stated complexity bound. □

9.8.2 Tractability and Approximations

Approximations are necessary for tractable implementation in complex models.

Definition 9.8.2 (Free Energy Approximation). An approximation $\tilde{F}_C$ of the true free energy F_C satisfies:

$$|F_C(s, o) - \tilde{F}_C(s, o)| \leq \epsilon(s, o)$$

for some error bound ϵ.

Theorem 9.8.2 (Approximation Impact). Using an approximation $\tilde{F}_C$ affects the gradient flow by at most:

$$\|\nabla_s F_C(s, o) - \nabla_s \tilde{F}_C(s, o)\| \leq \nabla_s \epsilon(s, o)$$

Proof: This follows directly from the properties of function approximation and gradient operations. □

9.9 Convergence and Optimization

9.9.1 Convergence of Case-Specific Learning

Learning in case-bearing models requires case-specific convergence analysis.

Definition 9.9.1 (Case-Specific Learning). A learning algorithm for case C generates a sequence of states $\{s_t\}$ such that:

$$s_{t+1} = s_t - \eta_t \nabla_s L_C(s_t, o)$$

where L_C is a case-specific loss function and η_t is the learning rate.

Theorem 9.9.1 (Convergence Conditions). The case-specific learning algorithm converges if: 1. L_C is μ-strongly convex 2. $\nabla_s L_C$ is L-Lipschitz continuous 3. The learning rate satisfies $0 < \eta_t < \frac{2}{\mu + L}$

Proof: Under these conditions, the optimization algorithm is a contraction mapping in the parameter space, guaranteeing convergence to the unique minimum. □

9.9.2 Multi-Case Optimization

Optimizing across multiple cases requires balancing case-specific objectives.

Definition 9.9.2 (Multi-Case Objective). A multi-case objective function combines case-specific objectives with weights α_C:

$$L(\{s_C\}) = \sum_{C \in \mathcal{C}} \alpha_C L_C(s_C, o)$$

where $\{s_C\}$ is the collection of case-specific states.

Theorem 9.9.2 (Pareto Optimality). A state collection $\{s_C^*\}$ is Pareto optimal if there exists no alternative collection $\{s_C'\}$ such that $L_C(s_C', o) \leq L_C(s_C^*, o)$ for all C with strict inequality for at least one case.

Proof: This follows from the definition of Pareto optimality in multi-objective optimization. □

9.10 Formal Correctness and Verification

9.10.1 Type Theory for Cases

Type theory provides a foundation for formal verification of case operations.

Definition 9.10.1 (Case Type System). A type system for cases assigns to each case C a type T_C representing its interface, with case transformations $f : C_1 \to C_2$ corresponding to type morphisms $T_f : T_{C_1} \to T_{C_2}$.

Theorem 9.10.1 (Type Safety). A well-typed case operation cannot result in interface mismatches or precision violations.

Proof: By the properties of a sound type system, well-typed terms do not "go wrong" during evaluation. The case type system ensures that operations only access interface elements that exist and respect precision constraints. □

9.10.2 Invariants and Properties

Formal verification ensures certain properties of case-bearing models.

Definition 9.10.2 (Case Invariant). A case invariant I_C is a property that holds for all valid states s of a model in case C: $I_C(s) = \text{true}$.

Theorem 9.10.2 (Invariant Preservation). For a transformation $f : C_1 \to C_2$ and invariants I_{C_1} and I_{C_2}, if $I_{C_1}(s) \implies I_{C_2}(f(s))$ for all s, then the transformation preserves invariants.

Proof: This follows directly from the definition of invariant preservation under transformation. □

References

1. Amari, S. I. (2016). Information geometry and its applications. Springer.
2. Friston, K. (2010). The free-energy principle: a unified brain theory? Nature Reviews Neuroscience, 11(2), 127-138.
3. Carlsson, G. (2009). Topology and data. Bulletin of the American Mathematical Society, 46(2), 255-308.
4. Mac Lane, S. (2013). Categories for the working mathematician. Springer.
5. Friston, K., FitzGerald, T., Rigoli, F., Schwartenbeck, P., & Pezzulo, G. (2017). Active inference: a process theory. Neural Computation, 29(1), 1-49.

6. Pierce, B. C. (2002). Types and programming languages. MIT press.
7. Guckenheimer, J., & Holmes, P. (2013). Nonlinear oscillations, dynamical systems, and bifurcations of vector fields. Springer.
8. Boyd, S., & Vandenberghe, L. (2004). Convex optimization. Cambridge university press.
9. Parisi, G. (1988). Statistical field theory. Addison-Wesley.
10. Spivak, D. I. (2014). Category theory for the sciences. MIT Press.

Supplement 10: Algorithmic Details & Pseudocode

This supplement provides detailed, language-agnostic pseudocode for key algorithms within the CEREBRUM framework to aid understanding, implementation, and reproducibility.

10.1 Core `CaseModel` Representation

The fundamental data structure in CEREBRUM is the `CaseModel`, which encapsulates the state, parameters, interfaces, and case-specific behaviors of a model.

```
class CaseModel:
    // Core properties
    State s                  // Current belief state
    Parameters θ             // Model parameters
    Case currentCase         // Current case assignment (NOM, ACC, DAT, etc.)
    PrecisionParameters π    // Precision weights for different model components
    InterfaceMap interfaces  // Available input/output interfaces

    // Core methods
    function transform(targetCase)              // Transform to a different case
    function calculateFreeEnergy()              // Compute current free energy
   function updateBeliefs(observation)       // Update beliefs based on observation
  function generatePrediction()           // Generate predictions from current state
  function selectOptimalCase(context)      // Select optimal case for current context
```

The `CaseModel` structure includes both common methods applicable to all cases and specialized methods that become active based on the current case assignment.

10.2 Case Transformation Algorithm

The `transform` method changes a model from its current case to a target case, adapting its interfaces, precision weights, and operational characteristics.

```
function transform(model, targetCase):
    // Input: Current model, target case
    // Output: Transformed model

    // 1. Verify transformation validity
    if not isValidTransformation(model.currentCase, targetCase):
        throw InvalidTransformationError

    // 2. Create new parameter mapping based on transformation type
    newParameters = mapParameters(model.θ, model.currentCase, targetCase)

    // 3. Adjust precision weights according to target case requirements
    newPrecision = mapPrecision(model.π, model.currentCase, targetCase)

    // 4. Reconfigure input/output interfaces
    newInterfaces = configureInterfaces(targetCase)

    // 5. Update state representation if necessary
    newState = adaptState(model.s, model.currentCase, targetCase)

    // 6. Create transformed model with new configuration
    transformedModel = new CaseModel(
```

```
        state: newState,
        parameters: newParameters,
        currentCase: targetCase,
        precision: newPrecision,
        interfaces: newInterfaces
    )

    // 7. Verify coherence of the transformed model
    if not verifyCoherence(transformedModel):
        throw IncoherentTransformationError

    return transformedModel
```

10.2.1 Parameter Mapping Function

The parameter mapping function translates parameters between cases, preserving relevant information while adapting to the new case's requirements.

```
function mapParameters(parameters, sourceCase, targetCase):
    // Create parameter mapping based on case pair
    mapping = getParameterMapping(sourceCase, targetCase)

    // Initialize new parameters with defaults for target case
    newParameters = initializeDefaultParameters(targetCase)

    // Apply the mapping to transfer applicable parameters
    for each (sourceParam, targetParam) in mapping:
        newParameters[targetParam] = transformParameter(
            parameters[sourceParam],
            sourceCase,
            targetCase
        )

    return newParameters
```

10.2.2 Precision Remapping Function

The precision remapping function adjusts precision weights to match the emphasis appropriate for the target case.

```
function mapPrecision(precision, sourceCase, targetCase):
    // Create new precision structure for target case
    newPrecision = initializeDefaultPrecision(targetCase)

    // Apply case-specific transformations
    switch targetCase:
        case NOM:
            // Emphasize prediction accuracy
            newPrecision.likelihood = HIGH_PRECISION
            newPrecision.prior = MEDIUM_PRECISION
        case ACC:
            // Emphasize belief updating
            newPrecision.likelihood = MEDIUM_PRECISION
            newPrecision.prior = LOW_PRECISION
        case GEN:
```

```
            // Emphasize relationship parameters
            newPrecision.relationship = HIGH_PRECISION
            newPrecision.state = MEDIUM_PRECISION
        // ... other cases

    return newPrecision
```

10.2.3 Interface Configuration Function

The interface configuration function activates the appropriate input/output interfaces for the target case.

```
function configureInterfaces(targetCase):
    // Initialize empty interface map
    interfaces = new InterfaceMap()

    // Configure case-specific interfaces
    switch targetCase:
        case NOM:
            // Nominative case emphasizes outputs
            interfaces.addInterface("predictions", INTERFACE_OUTPUT)
            interfaces.addInterface("context", INTERFACE_INPUT, LOW_PRIORITY)
        case ACC:
            // Accusative case emphasizes inputs
            interfaces.addInterface("updates", INTERFACE_INPUT, HIGH_PRIORITY)
            interfaces.addInterface("state", INTERFACE_OUTPUT)
        case DAT:
            // Dative case acts as mediator
            interfaces.addInterface("indirectInput", INTERFACE_INPUT)
            interfaces.addInterface("goal", INTERFACE_OUTPUT)
        // ... other cases

    return interfaces
```

10.3 Free Energy Calculation Algorithm

The free energy calculation is central to CEREBRUM's operation. While the specific form varies by case (as detailed in Supplement 9), the general algorithm follows this structure:

```
function calculateFreeEnergy(model):
    // Dispatch to case-specific implementation
    switch model.currentCase:
        case NOM:
            return calculateNominativeFreeEnergy(model)
        case ACC:
            return calculateAccusativeFreeEnergy(model)
        // ... other cases

    throw UnsupportedCaseError
```

10.3.1 Nominative Case Free Energy

```
function calculateNominativeFreeEnergy(model):
    // Emphasizes prediction accuracy
```

```
    // Calculate complexity term (KL divergence from prior)
    complexity = calculateKLDivergence(model.s, model.prior)

    // Calculate accuracy term (expected log-likelihood)
    // Note the enhanced precision α > 1
    accuracy = model.π.likelihood * calculateExpectedLogLikelihood(model)

    // Free energy is complexity minus accuracy
    return complexity - accuracy
```

10.3.2 Accusative Case Free Energy

```
function calculateAccusativeFreeEnergy(model):
    // Emphasizes efficient belief updates

    // Calculate complexity term with reduced weight β < 1
    complexity = model.π.prior * calculateKLDivergence(model.s, model.prior)

    // Calculate accuracy term (expected log-likelihood)
    accuracy = calculateExpectedLogLikelihood(model)

    // Free energy is complexity minus accuracy
    return complexity - accuracy
```

10.4 Case Selection Algorithm

The case selection algorithm uses active inference principles to determine the optimal case for a model given the current context.

```
function selectOptimalCase(model, context):
    // 1. Define possible target cases to consider
    possibleCases = getValidTransformationTargets(model.currentCase)

    // 2. Calculate expected free energy for each possible case
    efe = {}
    for each targetCase in possibleCases:
        // Create hypothetical transformed model
        hypotheticalModel = simulateTransform(model, targetCase)

        // Calculate information gain (epistemic value)
        infoGain = calculateInformationGain(hypotheticalModel, context)

        // Calculate goal alignment (pragmatic value)
        goalAlignment = calculateGoalAlignment(hypotheticalModel, context)

        // Combine into expected free energy
        efe[targetCase] = infoGain + goalAlignment

    // 3. Apply softmax to get case selection probabilities
    probabilities = softmax(-efe, temperature)

    // 4. Select case (sampling or maximum)
    if explorationEnabled:
        selectedCase = sampleFromDistribution(probabilities)
```

```
    else:
        selectedCase = argmax(probabilities)

    return selectedCase
```

10.4.1 Information Gain Calculation

```
function calculateInformationGain(model, context):
    // Calculate mutual information between model state and future observations
    // I(future observations; model state | case)

    // Estimate expected posterior after observing context
    expectedPosterior = estimatePosterior(model, context)

    // Calculate KL from current posterior to expected posterior
    return calculateKLDivergence(expectedPosterior, model.s)
```

10.4.2 Goal Alignment Calculation

```
function calculateGoalAlignment(model, context):
    // Calculate how well the model in this case would achieve goals

    // Get goal-oriented preferences (could be from context or model)
    preferences = extractPreferences(context)

    // Calculate expected log probability of preferred outcomes
    expectedLogProb = 0
    for each outcome in possibleOutcomes:
        outcomeProbability = predictOutcome(model, outcome)
        expectedLogProb += outcomeProbability * log(preferences[outcome])

    return expectedLogProb
```

10.5 Multiple Dispatch Mechanism

CEREBRUM uses case-based multiple dispatch to route operations to appropriate implementation based on the model's current case.

```
// Case-based dispatch system

// Registry for case-specific handlers
dispatchRegistry = {}

// Registration function
function registerHandler(case, operation, handler):
    if not dispatchRegistry[operation]:
        dispatchRegistry[operation] = {}
    dispatchRegistry[operation][case] = handler

// Dispatch function
function dispatch(model, operation, ...args):
   if not dispatchRegistry[operation] or not dispatchRegistry[operation][model.currentCase]:
        throw UnsupportedOperationError("Operation not supported for this case")
```

```
    // Get the case-specific handler
    handler = dispatchRegistry[operation][model.currentCase]

    // Execute with the model and additional arguments
    return handler(model, ...args)

// Example handler registrations
registerHandler(NOM, "process", processAsNominative)
registerHandler(ACC, "process", processAsAccusative)
registerHandler(DAT, "process", processAsDative)
// ... etc.

// Usage
function process(model, data):
    return dispatch(model, "process", data)
```

10.6 Novel Case Algorithms

The novel cases introduced in Supplement 2 require specialized algorithms to implement their unique behaviors.

10.6.1 Conjunctive Case [CNJ] Algorithm

```
function synthesizeJointPrediction(models):
    // Input: List of models to integrate
    // Output: Joint prediction

    // 1. Extract individual predictions
    predictions = []
    weights = []
    for each model in models:
        predictions.append(model.generatePrediction())
        // Weight based on precision and reliability
        weights.append(calculateModelWeight(model))

    // 2. Normalize weights
    weights = normalize(weights)

    // 3. Determine integration method based on prediction types
    if predictionsAreDistributions(predictions):
        // For probabilistic predictions, use product of experts
        jointPrediction = productOfExperts(predictions, weights)
    else:
        // For point predictions, use weighted combination
        jointPrediction = weightedCombination(predictions, weights)

    // 4. Check consistency of joint prediction
    consistencyScore = evaluateConsistency(jointPrediction, predictions)
    if consistencyScore < CONSISTENCY_THRESHOLD:
        // Apply consistency optimization
     jointPrediction = optimizeForConsistency(jointPrediction, predictions, weights)

    return jointPrediction
```

10.6.2 Recursive Case [REC] Algorithm

```
function applySelfTransformation(model, maxDepth=5):
    // Input: Model to recursively transform, maximum recursion depth
    // Output: Result of recursive self-application

    // 1. Base case: stop at maximum depth or convergence
    if maxDepth <= 0:
        return model

    // 2. Create a copy to transform
    workingModel = copyModel(model)

    // 3. Apply model to itself
    transformedState = model.applyTo(workingModel.s)
    workingModel.s = transformedState

    // 4. Check for convergence
    if isConverged(workingModel.s, model.s):
        return workingModel

    // 5. Recursive application with decreased depth
    return applySelfTransformation(workingModel, maxDepth - 1)
```

10.6.3 Metaphorical Case [MET] Algorithm

```
function mapStructure(sourceModel, targetDomain):
    // Input: Source model, target domain description
    // Output: Mapped model in target domain

    // 1. Extract structural elements from source model
    sourceStructure = extractStructuralElements(sourceModel)

    // 2. Identify potential mappings to target domain
    candidateMappings = identifyPotentialMappings(sourceStructure, targetDomain)

    // 3. Score mappings based on structural preservation
    for each mapping in candidateMappings:
     mapping.score = evaluateStructuralPreservation(mapping, sourceStructure, targetDomain)

    // 4. Select best mapping
    bestMapping = selectBestMapping(candidateMappings)

    // 5. Apply mapping to create new model in target domain
    mappedModel = applyMapping(sourceModel, bestMapping, targetDomain)

    // 6. Verify that key invariants are preserved
    if not verifyInvariants(sourceModel, mappedModel, bestMapping):
        // Adjust mapping to preserve critical invariants
        mappedModel = adjustMapping(mappedModel, sourceModel, bestMapping)

    return mappedModel
```

10.6.4 Explicative Case [EXP] Algorithm

```
function generateExplanation(model, abstractionLevel):
    // Input: Model to explain, desired abstraction level
    // Output: Human-interpretable explanation

    // 1. Extract relevant model components based on abstraction level
    relevantComponents = extractRelevantComponents(model, abstractionLevel)

    // 2. Compute feature attribution/importance scores
    attributions = computeAttributions(model, relevantComponents)

    // 3. Filter to most important features based on attribution
   significantFeatures = filterSignificantFeatures(attributions, SIGNIFICANCE_THRESHOLD)

    // 4. Map technical features to domain concepts
    conceptualMapping = mapFeaturesToConcepts(significantFeatures)

    // 5. Generate explanation text using conceptual mapping
   explanationStructure = structureExplanation(conceptualMapping, abstractionLevel)

    // 6. Validate explanation against model behavior
    if not validateExplanation(explanationStructure, model):
        // Refine explanation to resolve inconsistencies
        explanationStructure = refineExplanation(explanationStructure, model)

    // 7. Format explanation according to abstractionLevel
    explanation = formatExplanation(explanationStructure, abstractionLevel)

    return explanation
```

10.6.5 Diagnostic Case [DIA] Algorithm

```
function diagnoseModel(targetModel, testCases=null):
    // Input: Model to diagnose, optional specific test cases
    // Output: Diagnostic report

    // 1. If no test cases provided, generate appropriate test cases
    if not testCases:
        testCases = generateTestCases(targetModel)

    // 2. Run tests and collect results
    results = {}
    anomalies = []
    for each test in testCases:
        // Run test and compare to expected behavior
        actual = executeTest(targetModel, test.input)
        expected = test.expectedOutput
        results[test.id] = compareResults(actual, expected)

        // Track anomalies
        if isAnomalous(results[test.id]):
            anomalies.append({
                test: test,
```

```
            actual: actual,
            expected: expected,
            divergence: calculateDivergence(actual, expected)
        })

    // 3. Localize anomalies to specific model components
    if anomalies:
        componentAnalysis = localizeAnomalies(targetModel, anomalies)

        // 4. Generate diagnostic hypothesis
        diagnosticHypotheses = generateHypotheses(componentAnalysis)

        // 5. Rank hypotheses by likelihood
        rankedHypotheses = rankHypotheses(diagnosticHypotheses, anomalies)
    else:
        rankedHypotheses = []

    // 6. Compile diagnostic report
    report = {
        testResults: results,
        anomalies: anomalies,
        hypotheses: rankedHypotheses,
        modelHealth: calculateModelHealth(results)
    }

    return report
```

10.6.6 Orchestrative Case [ORC] Algorithm

```
function orchestrateModels(task, availableModels, resources):
    // Input: Task to accomplish, available models, resource constraints
    // Output: Orchestration plan

    // 1. Decompose task into subtasks
    subtasks = decomposeTask(task)

    // 2. Analyze model capabilities and match to subtasks
    modelCapabilities = analyzeCapabilities(availableModels)
    assignments = matchModelsToSubtasks(subtasks, modelCapabilities)

    // 3. Create dependency graph for subtasks
    dependencyGraph = createDependencyGraph(subtasks)

    // 4. Allocate resources based on priority and constraints
    resourceAllocation = allocateResources(assignments, resources)

    // 5. Create execution schedule
    schedule = scheduleExecution(assignments, dependencyGraph, resourceAllocation)

    // 6. Define coordination protocol
    protocol = defineCoordinationProtocol(availableModels, assignments)

    // 7. Create monitoring and error handling strategy
    errorHandling = defineErrorHandlingStrategy(assignments, dependencyGraph)
```

```
    // 8. Compile orchestration plan
    plan = {
        assignments: assignments,
        schedule: schedule,
        resourceAllocation: resourceAllocation,
        protocol: protocol,
        errorHandling: errorHandling
    }

    return plan
```

10.6.7 Generative Case [GEN] Algorithm

```
function generateInstance(latentVector, constraints):
    // Input: Point in latent space, constraints on generated instance
    // Output: Generated instance satisfying constraints

    // 1. Apply initial decoding from latent vector
    candidate = decodeFromLatentSpace(latentVector)

    // 2. Evaluate constraint satisfaction
    constraintSatisfaction = evaluateConstraints(candidate, constraints)

    // 3. If constraints not satisfied, perform constrained optimization
    if not allConstraintsSatisfied(constraintSatisfaction):
        // Define constraint satisfaction objective
        objective = createConstraintObjective(constraints)

        // Perform gradient-based optimization in latent space
        optimizedLatent = optimizeLatentVector(latentVector, objective)

        // Decode optimized latent vector
        candidate = decodeFromLatentSpace(optimizedLatent)

    // 4. Apply post-processing to enhance quality
    candidate = postProcessInstance(candidate)

    // 5. Verify novelty (avoid duplicating training data)
    noveltyScore = assessNovelty(candidate)
    if noveltyScore < NOVELTY_THRESHOLD:
        // Adjust to increase novelty while maintaining coherence
        candidate = adjustForNovelty(candidate, noveltyScore)

    // 6. Final quality check
    qualityScore = assessQuality(candidate)
    if qualityScore < QUALITY_THRESHOLD:
        // Enhance quality through targeted refinement
        candidate = enhanceQuality(candidate)

    return candidate
```

10.7 Database Operations for Case-Bearing Models

The following pseudocode outlines common database operations for storing and retrieving case-bearing models.

10.7.1 Storing a Model

```
function storeModel(model, database):
    // 1. Serialize model core components
    serializedModel = {
        id: generateUniqueId(),
        case: model.currentCase,
        state: serializeState(model.s),
        parameters: serializeParameters(model.θ),
        precision: serializePrecision(model.π),
        metadata: {
            created: currentTimestamp(),
            version: MODEL_VERSION,
            description: model.metadata.description
        }
    }

    // 2. Store in database with appropriate indexing
    database.models.insert(serializedModel)

    // 3. Update case-specific indexes
    database.caseIndex.insert({
        case: model.currentCase,
        modelId: serializedModel.id,
        timestamp: currentTimestamp()
    })

    return serializedModel.id
```

10.7.2 Retrieving Models by Case

```
function getModelsByCase(case, database, limit=10):
    // 1. Query case index
    caseEntries = database.caseIndex.find({case: case})
                                    .sort({timestamp: -1})
                                    .limit(limit)

    // 2. Retrieve full models
    modelIds = caseEntries.map(entry => entry.modelId)
    models = database.models.find({id: {$in: modelIds}})

    // 3. Deserialize models
    deserializedModels = []
    for each modelData in models:
        deserializedModels.append(deserializeModel(modelData))

    return deserializedModels
```

10.7.3 Recording a Case Transformation

```
function recordTransformation(sourceModelId, targetModelId, transformationType, database):
    // 1. Create transformation record
    transformation = {
        id: generateUniqueId(),
        sourceModelId: sourceModelId,
        targetModelId: targetModelId,
        transformationType: transformationType,
        timestamp: currentTimestamp(),
        metadata: {}
    }

    // 2. Store transformation record
    database.transformations.insert(transformation)

    // 3. Update model links
    database.models.update(
        {id: sourceModelId},
        {$push: {transformations: transformation.id}}
    )

    database.models.update(
        {id: targetModelId},
        {$set: {sourceTransformation: transformation.id}}
    )

    return transformation.id
```

10.8 Complexity Notes

The algorithms presented in this supplement have varying computational complexity, which is analyzed in detail in Supplement 7. Key complexity considerations include:

1. **Case Transformation**: O(P) where P is the number of parameters in the model
2. **Free Energy Calculation**: O(S) where S is the size of the state space
3. **Case Selection**: O(C·S) where C is the number of candidate cases and S is the state space size
4. **Conjunctive Integration**: $O(M^2)$ where M is the number of models being integrated
5. **Metaphorical Mapping**: $O(N^3)$ in the general case, where N is the number of structural elements being mapped
6. **Diagnostic Testing**: O(T·P) where T is the number of test cases and P is the number of model parameters

For practical implementations, approximation techniques described in Supplement 7 can be used to reduce these complexity bounds for large-scale models.

Supplement 11: Formal Definitions of Core Linguistic Cases

This supplement provides rigorous mathematical and operational definitions for the traditional linguistic cases as implemented within the CEREBRUM framework.

11.1 Introduction

The CEREBRUM framework adopts and adapts traditional linguistic case systems as a foundational organizing principle for model roles and transformations. This supplement formalizes each core case, detailing its semantic origin, computational interpretation, mathematical formulation, and operational characteristics within the CEREBRUM context.

The formal definitions presented here complement the novel cases described in Supplement 2, providing a complete picture of the case system underpinning CEREBRUM. Each case definition includes precise mathematical formulations of free energy components, specifying how precision weighting and message passing operate in models assigned to that case.

11.2 Nominative Case [NOM]

11.2.1 Semantic Role

Agent, performer, experiencer, or primary subject of an action or state.

11.2.2 Computational Role

Prediction generation, model source, primary generative process.

11.2.3 Formal Definition

A model M in nominative case [NOM] is characterized by the following free energy formulation:

$$F_{NOM}[q] = D_{KL}[q(s)||p(s)] - \alpha_{NOM} \cdot E_q[\log p(o|s)]$$

With precision weighting that emphasizes generative accuracy:

$$\alpha_{NOM} > 1$$

The message passing operations prioritize forward prediction over belief updates:

$$\nabla_q F_{NOM} \approx -\alpha_{NOM} \cdot \nabla_q E_q[\log p(o|s)] + \nabla_q D_{KL}[q(s)||p(s)]$$

11.2.4 Expected Input/Output Signature

- **Input:** Context information, cues for prediction generation
- **Output:** Predictions, generated observations, forward model outcomes

11.2.5 Operational Definition

A model operates in NOM case when: - It functions as a primary source of predictions or state representations - It actively generates outputs based on its current belief state - Its interface emphasizes outward information flow - It maintains relatively stable internal states rather than rapidly updating to external inputs

11.2.6 Implementation Requirements

- Forward-pass dominant architecture with strong generative components
- Relatively higher computational allocation to prediction generation than belief updating
- Stable prior distributions that resist rapid modification
- Parameter configurations that favor output precision over input sensitivity

11.3 Accusative Case [ACC]

11.3.1 Semantic Role

Patient, direct object, or entity directly affected by an action.

11.3.2 Computational Role

Receiving updates, target of belief revision, state modification recipient.

11.3.3 Formal Definition

A model M in accusative case [ACC] is characterized by the following free energy formulation:

$$F_{ACC}[q] = \beta_{ACC} \cdot D_{KL}[q(s)||p(s)] - E_q[\log p(o|s)]$$

With precision weighting that emphasizes efficient belief updates:

$$\beta_{ACC} < 1$$

The message passing operations prioritize rapid belief updating based on new information:

$$\nabla_q F_{ACC} \approx -\nabla_q E_q[\log p(o|s)] + \beta_{ACC} \cdot \nabla_q D_{KL}[q(s)||p(s)]$$

11.3.4 Expected Input/Output Signature

- **Input:** Direct updates, modification signals, belief revision information
- **Output:** Updated state representations, change confirmations

11.3.5 Operational Definition

A model operates in ACC case when: - It serves as the direct recipient of updates or modifications - Its primary function is to efficiently revise its internal state - It prioritizes integrating new information over maintaining prior beliefs - It exhibits parameter changes directly coupled to input signals

11.3.6 Implementation Requirements

- Backward-pass dominant architecture with efficient update mechanisms
- Parameter configurations that favor rapid learning over stability
- Higher learning rates or adaptation coefficients
- Mechanisms for rapidly incorporating new evidence into belief distributions

11.4 Genitive Case [GEN]

11.4.1 Semantic Role

Possessor, source, or entity indicating relationship or association.

11.4.2 Computational Role

Relationship representation, context provision, parameter source, hierarchical link.

11.4.3 Formal Definition

A model M in genitive case [GEN] is characterized by the following free energy formulation:

$$F_{GEN}[q] = D_{KL}[q(s,r)||p(s,r)] - E_q[\log p(o|s,r)]$$

Where r represents explicit relationship parameters linking the model to other entities. The GEN-specific message passing emphasizes relationship maintenance:

$$\nabla_r F_{GEN} = \nabla_r D_{KL}[q(s,r)||p(s,r)] - \nabla_r E_q[\log p(o|s,r)]$$

With high precision assigned to relationship parameters:

$$\pi_{GEN}(r) > \pi_{GEN}(s)$$

11.4.4 Expected Input/Output Signature

- **Input:** Relationship queries, context requests, parameter requests
- **Output:** Associated parameters, contextual information, relationship specifications

11.4.5 Operational Definition

A model operates in GEN case when: - It functions as a source of parameters or context for other models - It maintains and provides information about relationships between entities - It responds to queries about associated elements or hierarchical structures - Its primary value comes from the relationships it represents rather than direct actions

11.4.6 Implementation Requirements

- Relational database-like structures for storing associative information
- Graph-based representations with explicit edge parameterization
- Parameter-sharing mechanisms for linked model components
- Optimization objectives that prioritize relational consistency

11.5 Dative Case [DAT]

11.5.1 Semantic Role

Indirect object, recipient, beneficiary, or goal of an action.

11.5.2 Computational Role

Information destination, indirect update recipient, interaction goal, routing endpoint.

11.5.3 Formal Definition

A model M in dative case [DAT] is characterized by the following free energy formulation:

$$F_{DAT}[q] = D_{KL}[q(s,g)||p(s,g)] - E_q[\log p(o|s,g)]$$

Where g represents goal parameters that shape how information is received and processed. The precision weighting balances state updates against goal alignment:

$$\pi_{DAT}(g) \approx \pi_{DAT}(s)$$

The update equations specifically incorporate goal-directed processing:

$$\nabla_q F_{DAT} = \nabla_q D_{KL}[q(s,g)||p(s,g)] - \nabla_q E_q[\log p(o|s,g)]$$

11.5.4 Expected Input/Output Signature

- **Input:** Indirect updates, goal-relevant information, mediated signals
- **Output:** Goal state information, transformation results, reception confirmations

11.5.5 Operational Definition

A model operates in DAT case when: - It serves as an indirect recipient of actions or information - It represents goals or intended outcomes of processes - It mediates or routes information rather than being a final endpoint - It maintains goal representations that influence how inputs are processed

11.5.6 Implementation Requirements

- Goal-conditioned processing pipelines
- Architectures with explicit goal parameter representations
- Information routing mechanisms based on goal states
- Optimization criteria that balance immediate updates with goal-directed outcomes

11.6 Instrumental Case [INS]

11.6.1 Semantic Role

Instrument, means, or tool used to accomplish an action.

11.6.2 Computational Role

Transformation application, function execution, process implementation, policy enactment.

11.6.3 Formal Definition

A model M in instrumental case [INS] is characterized by the following free energy formulation:

$$F_{INS}[q] = D_{KL}[q(s,a)||p(s,a)] - E_q[\log p(o|s,a)]$$

Where a represents action or transformation parameters. The precision weighting emphasizes effective action execution:

$$\pi_{INS}(a) > \pi_{INS}(s)$$

The gradient includes specific action-optimization terms:

$$\nabla_a F_{INS} = \nabla_a D_{KL}[q(s,a)||p(s,a)] - \nabla_a E_q[\log p(o|s,a)]$$

11.6.4 Expected Input/Output Signature

- **Input:** Function arguments, transformation specifications, process inputs
- **Output:** Transformed results, process outputs, action outcomes

11.6.5 Operational Definition

A model operates in INS case when: - It serves as a means to accomplish transformations or processes - It implements well-defined functions that map inputs to outputs - Its primary value comes from the transformations it performs - It maintains specialized parameters optimized for particular operations

11.6.6 Implementation Requirements

- Function composition architectures
- Specialized transformation layers or components
- Parameter optimization focused on transformation fidelity
- Input-output mapping with minimal state maintenance

11.7 Locative Case [LOC]

11.7.1 Semantic Role

Location, place, or position where an action occurs or state exists.

11.7.2 Computational Role

Context provision, environment representation, state container, reference frame.

11.7.3 Formal Definition

A model M in locative case [LOC] is characterized by the following free energy formulation:

$$F_{LOC}[q] = D_{KL}[q(s,e)||p(s,e)] - E_q[\log p(o|s,e)]$$

Where e represents environmental or contextual parameters. The precision weighting emphasizes contextual stability:

$$\pi_{LOC}(e) > \pi_{LOC}(s)$$

The update equations prioritize context maintenance:

$$\nabla_e F_{LOC} = \nabla_e D_{KL}[q(s,e)||p(s,e)] - \nabla_e E_q[\log p(o|s,e)]$$

11.7.4 Expected Input/Output Signature

- **Input:** Context queries, environmental parameter requests, situational information requests
- **Output:** Environmental parameters, contextual information, situational representations

11.7.5 Operational Definition

A model operates in LOC case when: - It serves as a representation of the environment or context - It provides stable background parameters for other processes - It maintains information about spatial, temporal, or conceptual frameworks - It responds to queries about the current operational context

11.7.6 Implementation Requirements

- Context-encoding architectures with high-dimensional state spaces
- Slow-changing parameter regimes for environmental stability
- Efficient query mechanisms for contextual information retrieval
- State representations that condition other models' operations

11.8 Ablative Case [ABL]

11.8.1 Semantic Role

Source, origin, cause, or starting point of movement or action.

11.8.2 Computational Role

Information origin, causal source, initialization provider, trigger point.

11.8.3 Formal Definition

A model M in ablative case [ABL] is characterized by the following free energy formulation:

$$F_{ABL}[q] = D_{KL}[q(s,o)||p(s,o)] - E_q[\log p(o'|s,o)]$$

Where o represents origin parameters and o' represents downstream observations. The precision weighting emphasizes source accuracy:

$$\pi_{ABL}(o) > \pi_{ABL}(s)$$

The source-specific gradient focuses on causal origin representation:

$$\nabla_o F_{ABL} = \nabla_o D_{KL}[q(s,o)||p(s,o)] - \nabla_o E_q[\log p(o'|s,o)]$$

11.8.4 Expected Input/Output Signature

- **Input:** Triggering signals, initialization requests, source queries
- **Output:** Initial values, originating information, causal indicators

11.8.5 Operational Definition

A model operates in ABL case when: - It functions as a source of causal processes or information flows - It provides initialization values for other processes - It represents the origin point of transformations or sequences - It maintains and provides information about where processes began

11.8.6 Implementation Requirements

- Initialization parameter storage with high precision
- Causal tracking mechanisms for process origins
- Architectural components for transformative triggering
- Event-origin association mechanisms

11.9 Vocative Case [VOC]

11.9.1 Semantic Role

Addressee, entity being directly called or addressed.

11.9.2 Computational Role

Interface activation, attention target, communication endpoint, selective listener.

11.9.3 Formal Definition

A model M in vocative case [VOC] is characterized by the following free energy formulation:

$$F_{VOC}[q] = D_{KL}[q(s,c)||p(s,c)] - E_q[\log p(o|s,c)]$$

Where c represents communication or attention parameters. The precision weighting emphasizes selective attention:

$$\pi_{VOC}(c) > \pi_{VOC}(s)$$

The attention-specific update equations highlight selective information processing:

$$\nabla_c F_{VOC} = \nabla_c D_{KL}[q(s,c)||p(s,c)] - \nabla_c E_q[\log p(o|s,c)]$$

11.9.4 Expected Input/Output Signature

- **Input:** Attention signals, activation cues, addressing patterns
- **Output:** Availability indicators, attention confirmation, readiness signals

11.9.5 Operational Definition

A model operates in VOC case when: - It selectively attends to communication directed specifically to it - It serves as an explicit target for interaction - It acts as a communication endpoint rather than a mediator - It implements activation gates that control when it processes input

11.9.6 Implementation Requirements

- Attention mechanisms with selective filtering
- Activation gate architectures with identity-based triggers
- Signal detection components for targeted addressing
- Computational efficiency through selective processing

11.10 Case Relationships and Transformations

The core cases defined above form a structured system of functional roles that models can assume within the CEREBRUM framework. These cases are not independent but rather form a coherent system with well-defined transformation paths between them.

11.10.1 Common Case Transformations

Source Case	Target Case	Transformation Description	Primary Parameter Shifts
NOM → ACC	Prediction to Update	Model shifts from generating predictions to receiving updates	$\alpha_{NOM} \rightarrow \beta_{ACC}$ (precision shift from accuracy to complexity)
ACC → NOM	Update to Prediction	Model shifts from receiving updates to generating predictions	$\beta_{ACC} \rightarrow \alpha_{NOM}$ (precision shift from complexity to accuracy)
GEN → DAT	Relation to Goal	Model shifts from representing relationships to representing goals	$\pi_{GEN}(r) \rightarrow \pi_{DAT}(g)$ (precision shift from relationships to goals)
INS → LOC	Process to Context	Model shifts from implementing transformations to providing context	$\pi_{INS}(a) \rightarrow \pi_{LOC}(e)$ (precision shift from actions to environment)
ABL → INS	Source to Process	Model shifts from being an origin to implementing a process	$\pi_{ABL}(o) \rightarrow \pi_{INS}(a)$ (precision shift from origin to action)
VOC → NOM	Attention to Generation	Model shifts from receiving attention to generating content	$\pi_{VOC}(c) \rightarrow \alpha_{NOM}$ (precision shift from attention to accuracy)
LOC → GEN	Context to Relation	Model shifts from representing environment to representing relationships	$\pi_{LOC}(e) \rightarrow \pi_{GEN}(r)$ (precision shift from environment to relationships)

11.10.2 Case Properties Summary Table

Case	Primary Focus	Precision Emphasis	Interface Direction	Update Priority	Typical Role
NOM	Prediction generation	Accuracy ($\alpha_{NOM} > 1$)	Output-dominant	Stable beliefs	Generative model
ACC	Belief updating	Complexity ($\beta_{ACC} < 1$)	Input-dominant	Rapid updates	Update target
GEN	Relationship representation	Relationship parameters ($\pi_{GEN}(r)$)	Query-response	Relational consistency	Parameter source
DAT	Goal representation	Balanced ($\pi_{DAT}(g) \approx \pi_{DAT}(s)$)	Mediational	Goal-directed updates	Indirect recipient
INS	Action execution	Action parameters ($\pi_{INS}(a)$)	Transformational	Process optimization	Function implementation
LOC	Context provision	Environmental parameters ($\pi_{LOC}(e)$)	Contextual	Contextual stability	Environment representation
ABL	Source representation	Origin parameters ($\pi_{ABL}(o)$)	Initiating	Origin accuracy	Causal source
VOC	Selective attention	Communication parameters ($\pi_{VOC}(c)$)	Communicational	Attention gating	Interaction target

11.10.3 Computational Implications of Case Assignment

The assignment of a model to a specific case has significant implications for its computational behavior:

1. **Parameter Update Frequency**: NOM and LOC cases typically have slower parameter updates, while ACC cases update rapidly in response to inputs.
2. **Computational Resource Allocation**: Different cases require different computational resources - INS cases prioritize transformation efficiency, while GEN cases emphasize relationship storage and retrieval.
3. **Interface Design**: Case assignment shapes interface requirements - VOC cases need selective attention mechanisms, while DAT cases require goal-conditioned processing pipelines.
4. **Composition Rules**: Cases determine how models can be composed - NOM cases typically feed into ACC cases, while GEN cases provide parameters to other models.
5. **Error Handling**: Each case has characteristic error modes - NOM cases may suffer from generative inaccuracy, while ACC cases might exhibit belief instability.

This comprehensive case system provides a principled foundation for the design and implementation of CEREBRUM models, ensuring consistent functional roles across the framework.

Supplement 12: Writing a Research Paper

This supplement provides comprehensive guidelines for applying CEREBRUM methodologies to the research paper writing process, detailing how linguistic case frameworks can enhance scientific communication and knowledge representation.

12.1 Introduction

The research paper writing process presents a complex cognitive challenge that can be systematically addressed using CEREBRUM's linguistic case frameworks. This supplement outlines how viewing the various components and stages of research paper development through the lens of CEREBRUM's case-based approach can enhance organization, clarity, and knowledge integration throughout the writing process.

12.2 Applying CEREBRUM Cases to Research Paper Components

12.2.1 Mapping Components to Cases

Each section of a research paper can be conceptualized through specific linguistic cases, enabling more effective planning and execution:

Table S12.1: Research Paper Components Mapped to CEREBRUM Cases

Paper Component	Primary Case	Secondary Case	Functional Role
Abstract	NOM	ABL	Generates concise representation of overall work
Introduction	ABL	LOC	Establishes origin and contextual environment
Background	LOC	GEN	Provides contextual framework and relationships
Methods	INS	DAT	Details transformative processes and their goals
Results	ACC	NOM	Receives and presents outcomes of methods
Discussion	GEN	DAT	Establishes relationships between results and broader goals
Conclusion	DAT	VOC	Addresses field-level goals and audience
References	GEN	LOC	Defines relationships to external knowledge context

12.2.2 Key Benefits of Case-Based Approach

- **Structural Clarity**: Viewing each section through its case role clarifies its purpose and relationship to other sections

- **Functional Consistency**: Ensures each component fulfills its specific informational role
- **Integration Coherence**: Maintains cohesive relationships between paper components
- **Cognitive Organization**: Aligns writing process with natural information processing patterns
- **Targeted Precision**: Applies appropriate precision weighting to different research elements

12.3 The Research Paper Writing Process Through CEREBRUM

12.3.1 Pre-Writing Phase

- **[NOM]** Generate initial research questions and hypotheses
  - Apply generative processes to identify novel research directions
  - Maintain high precision on accuracy of fundamental claims
  - Develop predictive models that will be tested
- **[LOC]** Establish research context
  - Create environmental representations of the research field
  - Identify contextual parameters relevant to the work
  - Define boundary conditions and scope
- **[GEN]** Map relationships to existing literature
  - Establish associative links to foundational works
  - Define ownership of ideas and intellectual lineage
  - Structure hierarchical connections between concepts

12.3.2 Research Design Phase

- **[DAT]** Formulate research goals
  - Define specific objectives with measurable outcomes
  - Establish goal parameters that shape methodological choices
  - Balance competing research priorities
- **[INS]** Design methodological approach
  - Focus on transformation processes that generate knowledge
  - Optimize action parameters for experimental efficiency
  - Implement appropriate function composition for complex methods
- **[ABL]** Identify sources and starting points
  - Establish causal origins for research process
  - Define initialization parameters for experiments or analyses
  - Create clear trigger points for procedural sequences

12.3.3 Data Collection and Analysis Phase

- **[INS]** Execute data collection processes
  - Implement transformation functions with high fidelity
  - Maintain process documentation for reproducibility
  - Optimize action parameter precision
- **[ACC]** Integrate incoming results
  - Rapidly update beliefs based on experimental outcomes
  - Maintain appropriate uncertainty estimates
  - Prioritize efficient belief revision over stability
- **[GEN]** Establish relationships between data points
  - Create associative structures in result representations
  - Define hierarchical organization of findings
  - Map parameter relationships across experimental conditions

12.3.4 Writing Phase

- **[NOM]** Draft initial manuscript
  - Generate coherent narrative representations
  - Produce forward predictions about reader comprehension
  - Maintain stable belief structure throughout narrative
- **[INS]** Apply discipline-specific writing conventions
  - Execute stylistic transformations according to field requirements
  - Implement citation formatting and structural rules
  - Apply terminological consistency functions
- **[LOC]** Establish section-specific contexts
  - Create appropriate framing for each paper component
  - Maintain consistent environmental parameters within sections
  - Provide stable reference frames for complex concepts

12.3.5 Revision Phase

- **[ACC]** Incorporate feedback and revisions
  - Rapidly update manuscript based on reviewer input
  - Prioritize belief revision over original stability
  - Optimize for efficient integration of critical perspectives
- **[VOC]** Address specific reviewer concerns
  - Implement selective attention to particular critiques
  - Create targeted communication with specific audiences
  - Optimize response precision for critical issues
- **[GEN]** Strengthen relationship clarity
  - Enhance associative links between concepts
  - Clarify ownership of ideas and appropriate attribution
  - Reinforce hierarchical structure of argumentation

12.4 Precision Management in Research Paper Writing

The CEREBRUM framework's precision weighting approach provides valuable mechanisms for managing attention and resource allocation during the writing process:

Table S12.2: Precision Allocation in Research Paper Development

Writing Activity	Primary Precision Focus	Typical Weighting	Key Parameter
Literature Review	Relationship accuracy	High $\pi_{GEN}(r)$	Citation linkages
Methods Description	Process fidelity	High $\pi_{INS}(a)$	Procedural steps
Results Reporting	Data accuracy	High α_{NOM}	Statistical outcomes
Interpretation	Relationship plausibility	Balanced $\pi_{GEN}(r)$	Theoretical connections
Limitation Discussion	Uncertainty representation	Low β_{ACC}	Constraint identification
Conclusion	Goal relevance	High $\pi_{DAT}(g)$	Research impact

12.4.1 Strategic Precision Shifting

- **Early Drafts**: Lower precision on stylistic elements (INS case) while maintaining high precision on content accuracy (NOM case)
- **Revision Process**: Shift precision from content generation (NOM case) to feedback integration (ACC case)
- **Final Preparation**: Increase precision on communicative effectiveness (VOC case) and relationship clarity (GEN case)

12.5 Communication Message Passing in Scientific Writing

CEREBRUM's message passing formalism provides a framework for modeling how information flows through the scientific writing process:

12.5.1 Internal Message Passing

- **Author → Manuscript**: Information transfer from researcher beliefs to formal documentation
  - $\nabla_q F_{NOM} \approx -\alpha_{NOM} \cdot \nabla_q E_q[\log p(o|s)] + \nabla_q D_{KL}[q(s)||p(s)]$
  - High precision on minimizing divergence between author knowledge and manuscript representation
- **Section → Section**: Information coherence between paper components
  - $F_{GEN}[q] = D_{KL}[q(s,r)||p(s,r)] - E_q[\log p(o|s,r)]$
  - Emphasis on relationship parameters linking different paper sections

12.5.2 External Message Passing

- **Manuscript → Reader**: Knowledge transfer to scientific community
  - $F_{VOC}[q] = D_{KL}[q(s,c)||p(s,c)] - E_q[\log p(o|s,c)]$
  - Selective attention to communication parameters that enable effective transmission
- **Manuscript → Reviewer**: Targeted information for critical evaluation
  - $F_{DAT}[q] = D_{KL}[q(s,g)||p(s,g)] - E_q[\log p(o|s,g)]$
  - Goal-directed communication focused on satisfying scientific evaluation criteria

12.6 Practical CEREBRUM-Based Writing Strategies

12.6.1 Case-Transitional Writing Approaches

Table S12.3: Writing Transitions Between Cases

Transition	Strategy	Example Application
LOC → GEN	Context-to-Relationship	Moving from literature review to identifying research gaps
ABL → INS	Source-to-Process	Transitioning from research question to methodological approach
INS → ACC	Process-to-Outcome	Connecting methods to results
ACC → GEN	Outcome-to-Relationship	Interpreting results in theoretical context
GEN → DAT	Relationship-to-Goal	Connecting findings to research objectives

Transition	Strategy	Example Application
DAT → VOC	Goal-to-Audience	Translating research impact to field-specific implications

12.6.2 Writing Process Optimization

- **Parallelized Section Development**
  - Treat each section as a case-specific model operating semi-independently
  - Implement parallel processing for sections with different case assignments
  - Synchronize through explicit relationship parameters
- **Precision-Guided Revision Cycles**
  - Apply varying precision weights to different aspects during revision
  - Cycle through case-specific optimization in sequential editing passes
  - Gradually shift from content precision to communication precision
- **Goal-Directed Outlining**
  - Structure initial organization using DAT case formalism
  - Define explicit goal parameters for each section
  - Optimize section development to minimize goal-directed free energy
- **Relationship-Preserving Editing**
  - Maintain explicit tracking of cross-references during revision
  - Preserve relationship parameters when modifying connected sections
  - Implement consistency checks for relational integrity

12.7 Case-Based Solutions to Common Writing Challenges

- **Writer's Block [NOM dysfunction]**
  - Temporarily shift to ACC case to incorporate external inputs
  - Reduce precision requirements on initial content generation
  - Implement sampling-based approaches rather than maximum likelihood
- **Disorganized Structure [GEN dysfunction]**
  - Apply explicit relationship parameterization between sections
  - Increase precision on hierarchical organization
  - Implement graph-based representations of content flow
- **Unclear Methods [INS dysfunction]**
  - Increase precision on transformation parameters
  - Decompose complex procedures into sequential function applications
  - Optimize for process transparency rather than conciseness
- **Weak Discussion [GEN/DAT dysfunction]**
  - Enhance relationship precision between results and broader context
  - Explicitly parameterize goal relevance of findings
  - Balance precision between relationships and goals

12.8 Conclusion

The CEREBRUM framework provides a principled approach to the research paper writing process by mapping linguistic cases to functional components and processes. By conceptualizing the writing process through this lens, researchers can benefit from:

- Clearer structural organization aligned with cognitive processing patterns
- Systematic approaches to information flow management
- Precision-guided resource allocation during writing and revision
- Explicit treatment of relationships between paper components
- Formalized strategies for addressing common writing challenges

The case-based approach outlined in this supplement offers a comprehensive methodology for enhancing scientific communication through linguistically-informed cognitive principles.

12.9 Additional Tables for Research Paper Development

Table S12.4: CEREBRUM Case Application to Different Research Paper Types

Paper Type	Dominant Case Pattern	Optimization Focus	Key Advantages
Empirical Research	INS → ACC → GEN	Methodological rigor and data analysis	Clear process-to-outcome linkage
Theoretical Papers	NOM → GEN → DAT	Concept generation and relationship mapping	Strong theoretical framework development
Review Articles	LOC → GEN → DAT	Comprehensive context and relationship analysis	Effective knowledge synthesis
Case Studies	ACC → GEN → DAT	Observation detail and pattern recognition	Rich descriptive-to-interpretive flow
Methodological Papers	INS → INS → DAT	Process refinement and validation	Detailed technical specifications
Position Papers	NOM → DAT → VOC	Argument construction and audience engagement	Persuasive stance development
Interdisciplinary Research	GEN → MET → DAT	Cross-domain relationship mapping	Novel connection establishment

Table S12.5: Case-Specific Revision Strategies for Research Papers

Case Focus	Revision Strategy	Key Questions	Improvement Metrics
NOM	Prediction Clarity	Does the paper make clear, testable predictions?	Hypothesis specificity, predictive power
ACC	Data Presentation	Are results presented clearly and accurately?	Data visualization quality, statistical clarity
GEN	Relationship Coherence	Are connections between concepts logically structured?	Citation network coherence, argument flow
DAT	Goal Alignment	Does the work clearly connect to stated objectives?	Research question alignment, conclusion relevance
INS	Methodological Transparency	Are processes described with sufficient detail?	Reproducibility, methodological justification
LOC	Contextual Grounding	Is the work properly situated in its field?	Literature coverage, research gap identification

Case Focus	Revision Strategy	Key Questions	Improvement Metrics
ABL	Origin Clarity	Are starting points and motivations well-established?	Problem statement clarity, motivation strength
VOC	Audience Engagement	Does the paper effectively address its audience?	Readability, terminology appropriateness