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

Daniel Ari Friedman

Version 1.2 (2025-04-12)

Abstract

This paper introduces Case-Enabled Reasoning Engine with Bayesian Representations for Unified Modeling (CEREBRUM). CEREBRUM is a synthetic intelligence framework that integrates linguistic case systems with cognitive scientific principles to describe, design, and deploy generative models in an expressive fashion. By treating models as case-bearing entities that can play multiple contextual roles (e.g. like declinable nouns), CEREBRUM establishes a formal linguistic-type calculus for cognitive model use, relationships, and transformations. The CEREBRUM framework uses structures from category theory and modeling techniques related to the Free Energy Principle, in describing and utilizing models across contexts. CEREBRUM addresses the growing complexity in computational and cognitive modeling systems (e.g. generative, decentralized, agentic intelligences), by providing structured representations of model ecosystems that align with lexical ergonomics, scientific principles, and operational processes.

Contents

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 comprehensive approach to cognitive systems modeling by applying linguistic case systems to model management. This framework treats cognitive models as entities that can exist in different “cases”, as in a morphologically rich language, based on their functional role within an intelligence production workflow. This enables more structured representation of model relationships and transformations.

The code to generate this paper, and further open source development from are available at https://github.com/ActiveInferenceInstitute/CEREBRUM .

Introduction

Cognitive Systems Modeling

Cognitive systems modeling approaches cognition as a complex adaptive system, where cognitive processes emerge from the dynamic interaction of multiple components across different scales. This perspective draws from ecological psychology’s emphasis on organism-environment coupling, where cognitive processes are fundamentally situated in and shaped by their 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 are not merely internal representations but active participants in a broader cognitive ecosystem, where they adapt and evolve through interaction with other models and environmental constraints. This systems-level perspective is particularly relevant for intelligence production, where multiple analytical models must coordinate their activities 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 multiple interacting components that collectively produce intelligent behavior through their coordinated activity (including language use).

Active Inference

Active Inference is a first-principles account of perception, learning, and decision-making based on the Free Energy Principle. In this framework, cognitive systems minimize variational free energy bounded surprise, reflecting the difference between an organism’s internal model and its environment through perception (updating internal models) and action (changing action and ultimately sensory inputs). The Active Inference framework formalizes uncertainty in terms of entropy and precision weighting, enabling dynamic adaptive processes. While many model architectures are possible, hierarchical message passing is a common implementation that implements predictions as top-down flows and prediction errors as bottom-up flows, creating a bidirectional inference system that iteratively minimizes surprise across model levels. Active Inference treats all cognitive operations as Bayesian model update, providing a unifying mathematical formalism for predictive cognition.

Linguistic Case Systems

Linguistic case systems represent grammatical relationships between words through morphological marking. Case systems operate as morphosyntactic interfaces between semantics and syntax, encoding contextualized relationship types rather than just sequential ordering. This inherent relationality makes case systems powerful abstractions for modeling complex dependencies and transformations between conceptual entities. Cases under consideration here include nominative (subject), accusative (object), dative (recipient), genitive (possessor), instrumental (tool), locative (location), and ablative (origin), all serving different functional roles

within sentence structures. Languages implement these differently: nominative-accusative systems distinguish subjects from objects, while ergative-absolutive systems group intransitive subjects with direct objects. While English has largely lost its morphological case system, the underlying case relationships still exist and are expressed through word order and prepositions. For example, in "The cat chased the mouse," the nominative case is marked by position (subject before verb) rather than morphology, while in "I gave him the book," the dative case is marked by the preposition "to" (implied) and word order. This demonstrates that (the semantics/semiosis/pragmatics of) case relationships are fundamental to language structure, even when not overtly marked morphologically (e.g. expressed in writing or spoken language).

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. Modern implementations increasingly must 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.

Towards Languages for Generative Modeling

The Active Inference community has extensively explored numerous adjectival modifications of the base framework, including Deep, Affective, Branching-Time, Quantum, Mortal, Structured Inference, among others. Each adjectival-prefixed variant emphasizes specific architectural aspects or extensions of the core formalism. Building on this, CEREBRUM focuses on a wider range of linguistic formalism (e.g. in this paper, declensional semantics) rather than adjectival modifications. In this first CEREBRUM paper, there is an emphasis on the declensional aspects of generative models as noun-like entities, separate from adjectival qualification. This approach 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 and transposes structured modeling approaches to ecosystems of shared intelligence, while preserving the underlying (partitioned, flexible, variational, composable, interfacial, inter-active, empirical, applicable, communicable) semantics.

Conceptual Foundations: The Intersection of Four Domains

CEREBRUM integrates four key domains to create a unified framework for model management (Figure 1):

1. **Cognitive Systems Modeling** offers the entities that take on case relationships
2. **Active Inference** supplies the predictive processing mechanics that drive case transformations
3. **Linguistic Case Systems** provide the grammatical metaphor for how models relate to each other
4. **Intelligence Production** furnishes the practical application context and workflows

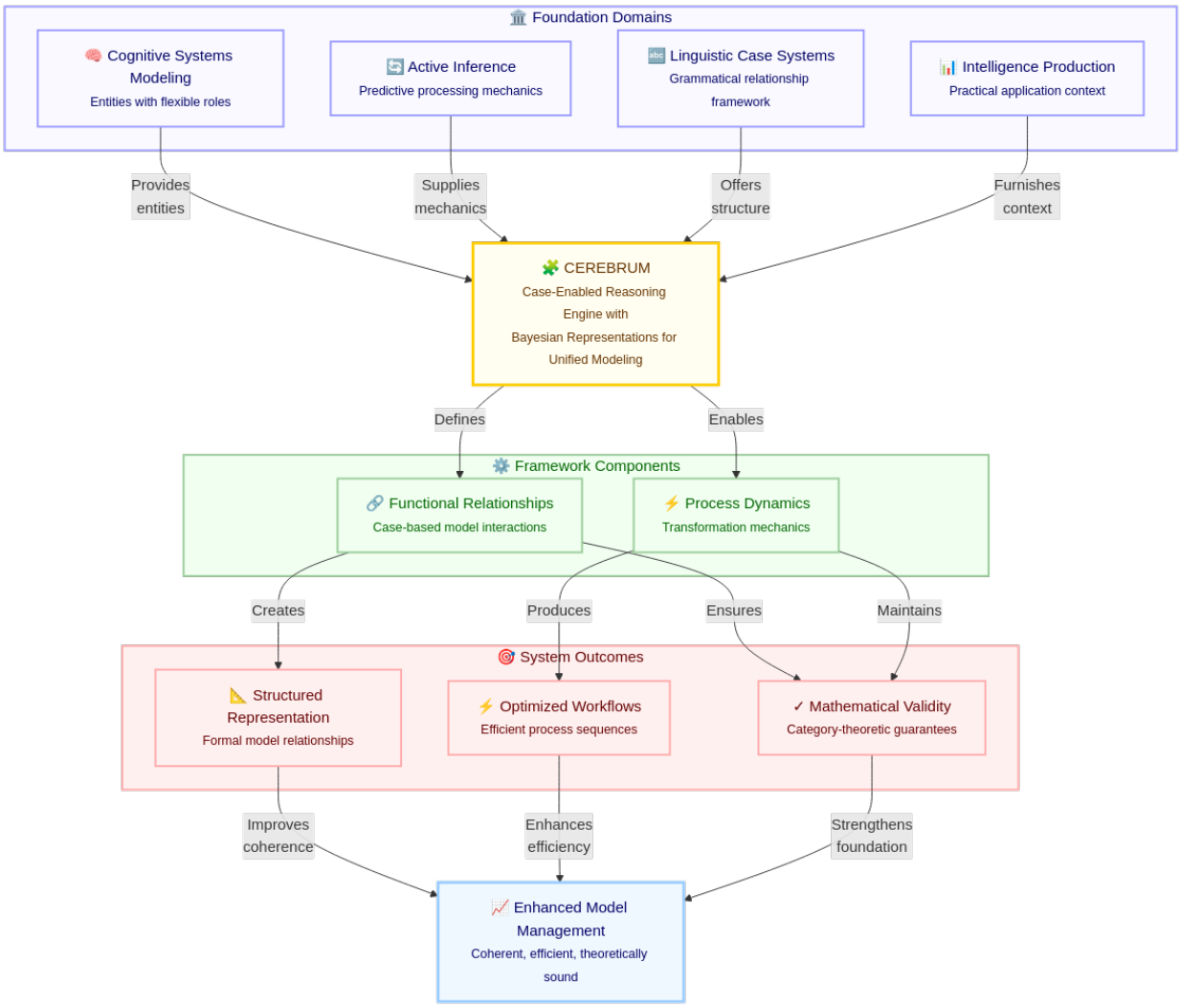


Figure 1: Figure 1. Foundation Domains of CEREBRUM. This diagram illustrates the conceptual architecture of the CEREBRUM framework, showing how four distinct domains converge to create a unified approach to model management. Cognitive Systems Modeling provides the entities that assume case relationships; 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.

Methods

Formal Framework Development

The CEREBRUM framework was developed as a part of a broader synthetic intelligence framework, combining 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. 2. **Category-Theoretic Mapping**: Implementing category theory to formalize morphisms between case states as functorial transformations. 3. **Algorithmic Implementation**: Developing algorithmic specifications for case transformations compliant with the Free Energy Principle. 4. **Variational Methods**: Applying variational free energy calculations to optimize model inference as well as structural transformations.

Mathematical Foundation

The mathematical foundation of CEREBRUM builds on formalizations of case transformations using category theory and variational inference. Case transformations are modeled as morphisms in a category where objects are 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.

Core Concept: Cognitive Models as Case-Bearing Entities

Just as nouns in morphologically rich languages take different forms based on their grammatical function, cognitive models in CEREBRUM can exist in different "states" or "cases" depending on how they relate to other models or processes within the system. Figure 2 illustrates this linguistic parallel.

The core framework of CEREBRUM organizes cognitive models according to their case relationships, as shown in Figure 3, which maps out the primary case assignments and their functional roles.

Case Functions in Cognitive Model Systems

Table 1: Case Functions in Cognitive Model Systems

Abbr	Case	Function in CEREBRUM	Example Usage
[NOM]	**Nominative**	Model as active agent; acts as the primary producer of predictions and exerts causal influence on other models	Model X [NOM] generates predictions about data distributions; controls downstream processing
[ACC]	**Accusative**	Model as object of process; receives transformations and updates from other models or processes	Process applies to Model X [ACC]; optimization procedures refine Model X's parameters
[GEN]	**Genitive**	Model as source/possessor; functions as the origin of outputs, products, and derived models	Output of Model X [GEN]; intelligence products derived from Model X's inferences

Abbr	Case	Function in CEREBRUM	Example Usage
[DAT]	**Dative**	Model as recipient; specifically configured to receive and process incoming data flows	Data fed into Model X [DAT]; Model X receives information from external sources
[INS]	**Instrumental**	Model as method/tool; serves as the means by which analytical operations are performed	Analysis performed via Model X [INS]; Model X implements analytical procedures
[LOC]	**Locative**	Model as context; provides environmental constraints and situational parameters	Parameters within Model X [LOC]; environmental contingencies modeled by X
[ABL]	**Ablative**	Model as origin/cause; represents historical conditions or causal precursors	Insights derived from Model X [ABL]; causal attributions traced to Model X
[VOC]	**Vocative**	Model as addressable entity; functions as a directly callable interface with name-based activation	"Hey Model X" [VOC]; direct invocation of Model X for task initialization; documentation reference point

Within intelligence production systems, these case relationships serve critical functional roles: nominative models act as primary analytical engines driving the intelligence case; accusative models become targets of quality assessment and improvement; multimodal generitive models generate documentation and reports; dative models receive and process collected intelligence data; instrumental models provide the methodological framework for investigations; locative models establish situational boundaries; ablative models represent the historical origins of analytical conclusions; and vocative models serve as directly addressable interfaces for command initiation and documentation reference. Together, these case relationships create a comprehensive framework for structured intelligence workflows. Figure 4 illustrates how this core framework integrates with intelligence case management.

A Preliminary Example of a Case-Bearing Model: Homeostatic Thermostat

Consider a cognitive model of a homeostatic thermostat that perceives room temperature with a thermometer, and regulates temperature through connected heating and cooling systems. In nominative case [NOM], the thermostat model actively generates temperature predictions and dispatches control signals, functioning as the primary agent in the temperature regulation process. When placed in accusative case [ACC], this same model becomes the object of optimization processes, with its parameters being updated based on prediction errors between expected and actual temperature readings. In dative case [DAT], the thermostat model receives environmental temperature data streams and occupant comfort preferences as inputs. The genitive case [GEN] transforms the model into a generator of temperature regulation reports and system performance analytics ("genitive AI"). When in instrumental case [INS], the thermostat serves as a computational tool implementing control algorithms for other systems requiring temperature management. The locative case [LOC] reconfigures the model to represent the contextual environment in which temperature regulation occurs,

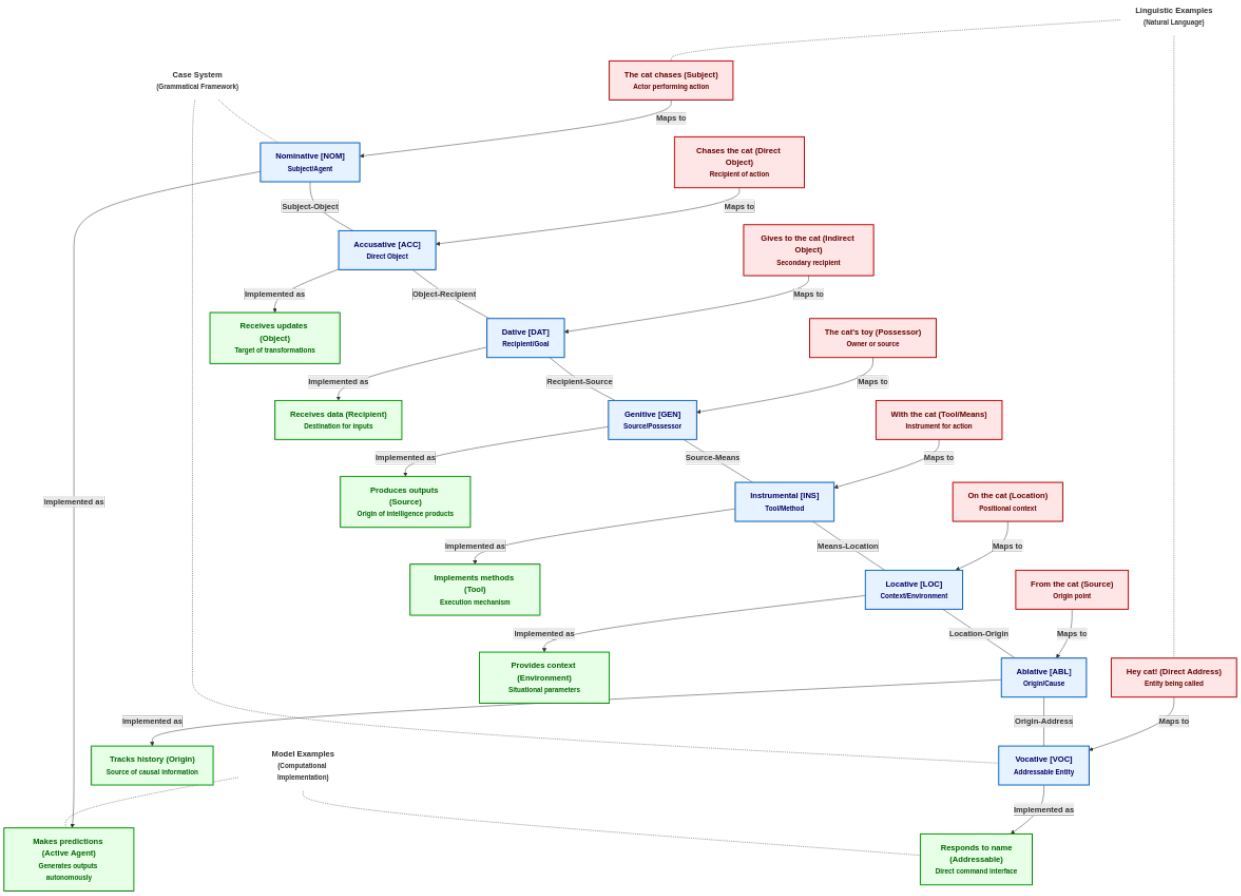


Figure 2: Figure 2. Case Relationships - Model and Linguistic Parallels. This figure illustrates the direct mapping between linguistic case systems and cognitive model relationships in CEREBRUM. The top row presents everyday linguistic examples of each case in English (though English largely expresses 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.

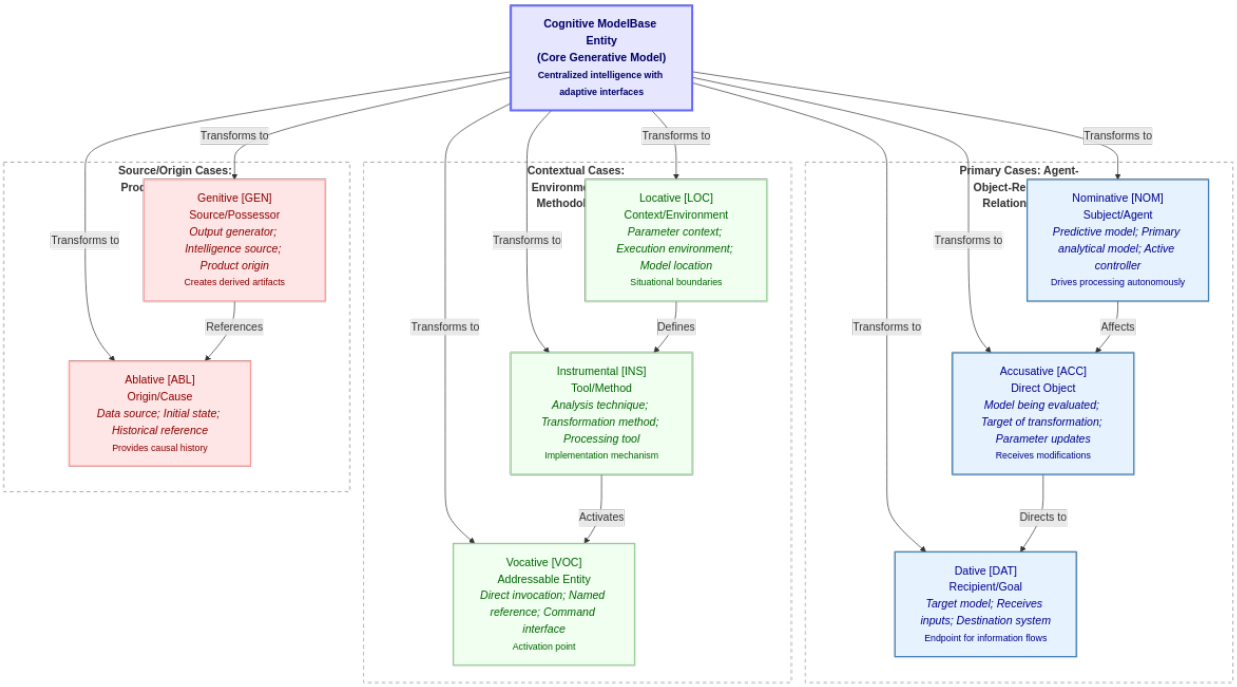


Figure 3: Figure 3. Cognitive Model Case Framework. This diagram illustrates how a single core generative model can assume different functional roles through case assignments. At the center lies the core cognitive model entity, which can be transformed into eight distinct case forms, each serving a specific 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.

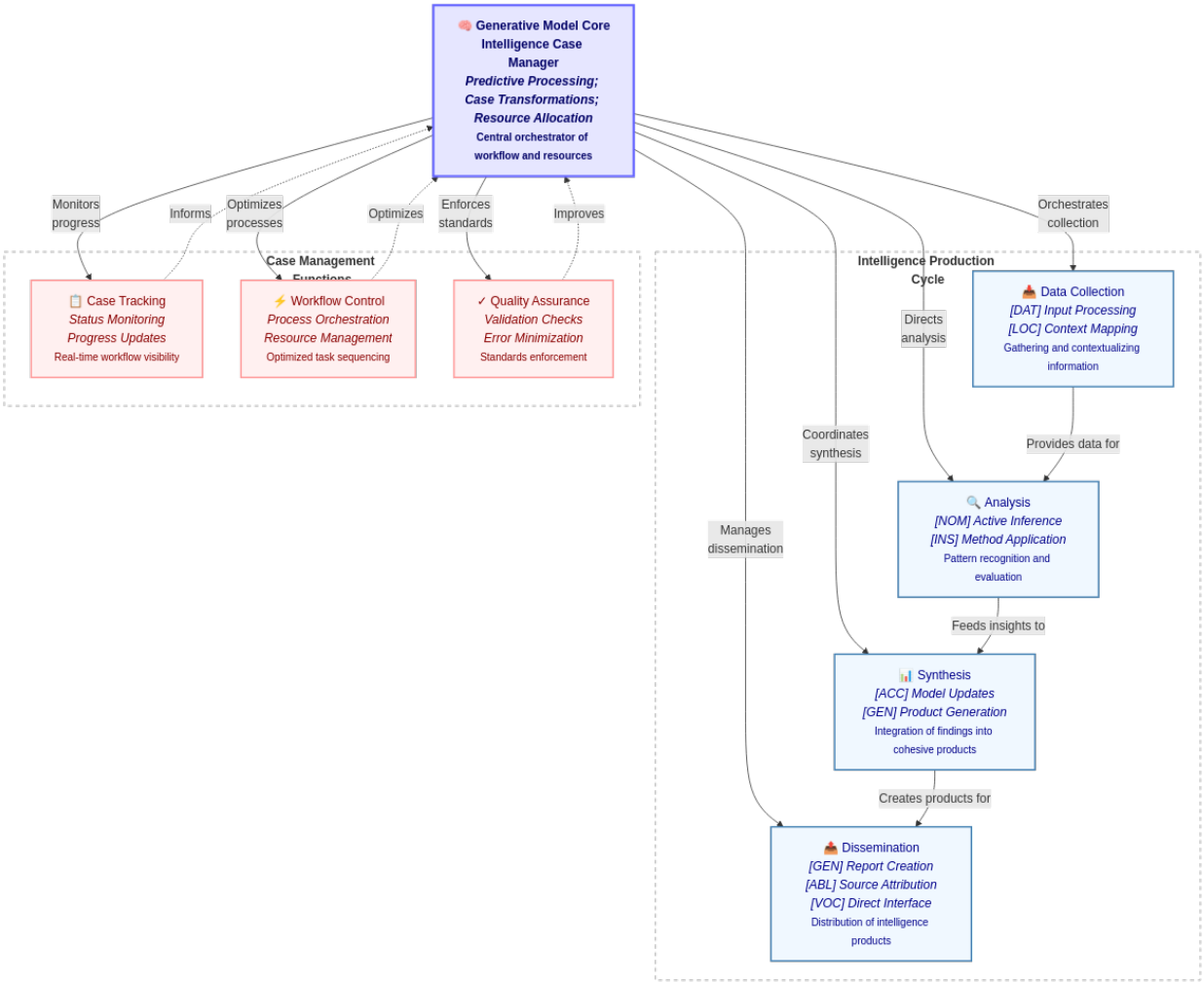


Figure 4: Figure 4. Generative Model Integration in Intelligence Case Management. This diagram illustrates how CEREBRUM's generative model core orchestrates both intelligence production and case management functions through case-specific transformations. At the center lies the core generative model that 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.

modeling building thermal properties, or discussing something within the model as a location. Finally, in ablative case [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 dramatically different functional roles while maintaining its core identity as a thermostat.

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, or variable roles defined by analytical pipelines, CEREBRUM treats cognitive models as flexible entities capable of morphological adaptation to different operational contexts.

Morphological Transformation of Generative Models

When an active inference generative model undergoes case transformation, it experiences orchestrated systematic changes such as: 1. **Functional Interfaces**: Input/output specifications change to match the case role requirements 2. **Parameter Access Patterns**: Which parameters are exposed or constrained changes based on case 3. **Prior Distributions**: Different cases employ different prior constraints on parameter values 4. **Update Dynamics**: The ways in which the model updates its internal states vary by case role 5. **Computational Resources**: Different cases receive different precision-weighted computational allocations

These changes 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]	Fully accessible parameters; all degrees of freedom available for prediction generation; strongest prior constraints on likelihood mapping	Outputs predictions; exposes forward inference pathways; prediction interfaces activated	Highest precision on likelihood; maximizes precision of generative mapping from internal states to observations
[ACC]	Restricted parameter access; plasticity gates opened; learning rate parameters prioritized	Receives transformations; update interfaces exposed; gradient reception pathways active	Highest precision on parameters; maximizes precision of parameter updates based on prediction errors
[DAT]	Input-focused parameterization; sensory mapping parameters prioritized; perceptual categorization parameters activated	Receives data flows; input processing interfaces exposed; sensory reception channels active	Highest precision on inputs; maximizes precision of incoming data relative to internal expectations

Case	Parametric Changes	Interface Transformations	Precision Weighting
[GEN]	“Genitive AI”; Output-focused parameterization; production parameters activated; generative pathway emphasis	Generates products; output interfaces prioritized; production pathways activated	Highest precision on outputs; maximizes precision of generated products relative to internal models
[INS]	Method-oriented parameters exposed; algorithmic parameters accessible; procedural knowledge emphasized	Implements processes; computational interfaces active; procedural execution pathways open	Highest precision on operations; maximizes precision of procedural execution relative to methodological expectations
[LOC]	Context parameters emphasized; environmental modeling parameters prioritized; situational knowledge emphasized	Provides environmental constraints; contextual interfaces active; environmental modeling pathways prioritized	Highest precision on contexts; maximizes precision of contextual representation relative to environmental dynamics
[ABL]	Origin states emphasized; historical parameters accessible; causal attribution pathways strengthened	Source of information; historical data interfaces active; causal explanation pathways open	Highest precision on historical data; maximizes precision of causal attributions and historical reconstructions
[VOC]	Identity parameters prioritized; naming and identification parameters activated; interface exposure emphasized	Maintains addressable interfaces; name recognition pathways activated; command reception channels open	Highest precision on identification cues; maximizes precision of name recognition relative to calling patterns

Active Inference Model Declension

Consider a perception-oriented generative model M with parameters theta, internal states s, and observational distribution p(o|s,theta). When declined across cases, this single model transforms as follows: - **M[NOM]**: Actively generates predictions by sampling from p(o|s,theta), 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] represents a unique functional role where models serve as directly addressable entities within a model ecosystem. Unlike other cases that focus on data processing or transformational aspects, the vocative case specifically optimizes a model for name-based recognition and command reception.

This has particular relevance in synthetic intelligence environments where models must be selectively activated or “woken up” through explicit address, similar to how humans are called by name to gain their attention. The vocative case maintains specialized interfaces for handling direct commands, documentation references, and initialization requests. In practical applications, models in vocative case might serve as conversational agents awaiting activation, documentation reference points within technical specifications, or system components that remain dormant until explicitly addressed. This pattern mimics the linguistic vocative case where a noun is used in direct address, as in “Hey Siri” or “OK Google” activation phrases for digital assistants, creating a natural bridging pattern 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 their core identity.

A sequence diagram illustrating the workflow of case transformations is shown in Figure 5.

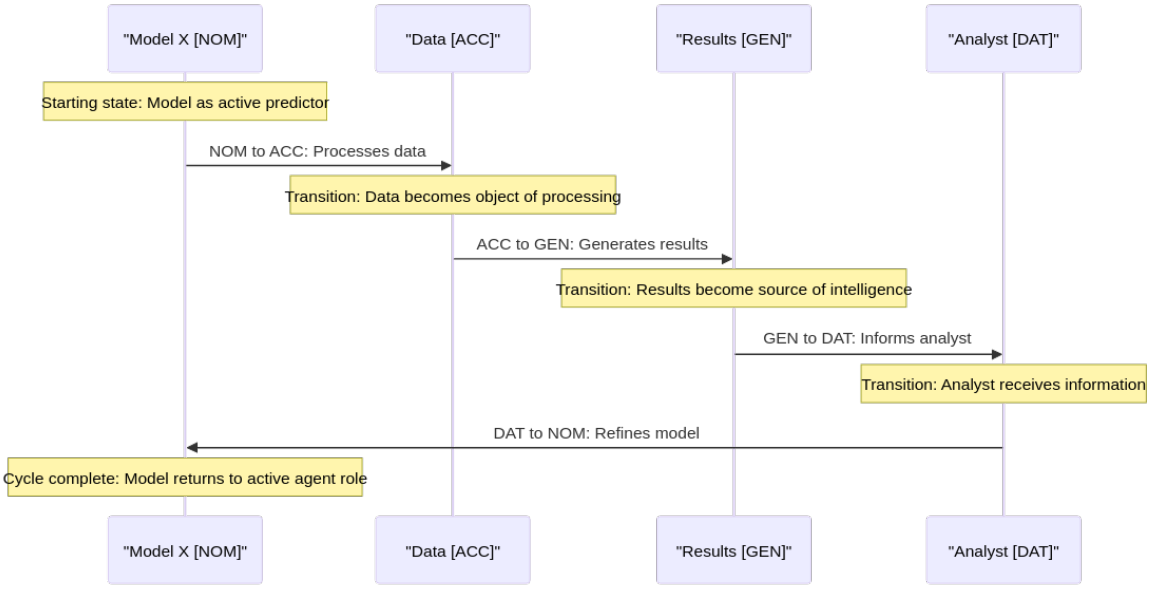


Figure 5: Figure 5. Model Workflows as Case Transformations - Sequence Diagram. This diagram illustrates the cyclical nature of case transformations in an intelligence workflow. The sequence begins with Model X in Nominative case [NOM] functioning as an active agent that processes data (in Accusative 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 supports different alignment systems for model relationships, mirroring linguistic morphosyntactic structures. These alignment patterns determine how models interact and transform based on their functional roles.

Figure 9 illustrates the core alignment patterns derived from linguistic theory, showing how models can be organized based on their case relationships. This includes nominative-accusative alignment (where models are distinguished by their role as agents or patients), ergative-absolutive alignment (where models are grouped by their relationship to actions), and tripartite alignment (where each case is marked distinctly).

Figure 10 demonstrates the practical implementation of these alignment patterns in model ecosystems. The diagram illustrates the computational implications of each alignment pattern, including resource allocation, message passing, and transformation efficiency. This implementation view complements the theoretical alignment patterns shown in Figure 9 by demonstrating their practical application in cognitive model management.

Implementation in Intelligence Production

As previously discussed (see Figure 4 and Figure 6), CEREBRUM integrates case transformations into intelligence production workflows. Figure 11 and Figure 12 provide alternative state-based visualizations illustrating how models transition between cases during the intelligence lifecycle.

Category theory provides a formal mathematical foundation for understanding these case transformations. Figure 7 and Figure 8 illustrate two complementary perspectives on the category-theoretic framework underlying CEREBRUM.

The intelligence production workflow, as managed by CEREBRUM, begins with raw data collection. In this initial phase, models assigned the instrumental case [INS] serve as data collection tools, implementing specific methods for information gathering.

As data moves to the preprocessing stage, models transition to the nominative case [NOM]. In this role, they become active agents, performing tasks such as cleaning, normalizing, and preparing the data for analysis.

During the analysis phase, models assume the locative case [LOC]. Here, they provide essential contextual understanding and define the environmental parameters that shape the analytical process, establishing the setting or conditions for the analysis.

Integration represents a critical transition point. Models in the genitive case [GEN] are responsible for generating intelligence products by synthesizing information gathered from multiple sources and previous stages.

These generated products then undergo evaluation. Models assigned the accusative case [ACC] perform this function, acting as the objects of assessment processes. They are evaluated for quality, accuracy, and relevance, identifying areas requiring improvement.

The refinement phase employs models in the dative case [DAT]. These models are configured to receive feedback from the evaluation stage and implement necessary changes or adjustments to the intelligence products or the underlying analysis.

Finally, in the deployment stage, models often return to the nominative case [NOM] for the active implementation or dissemination of the refined intelligence solutions or products.

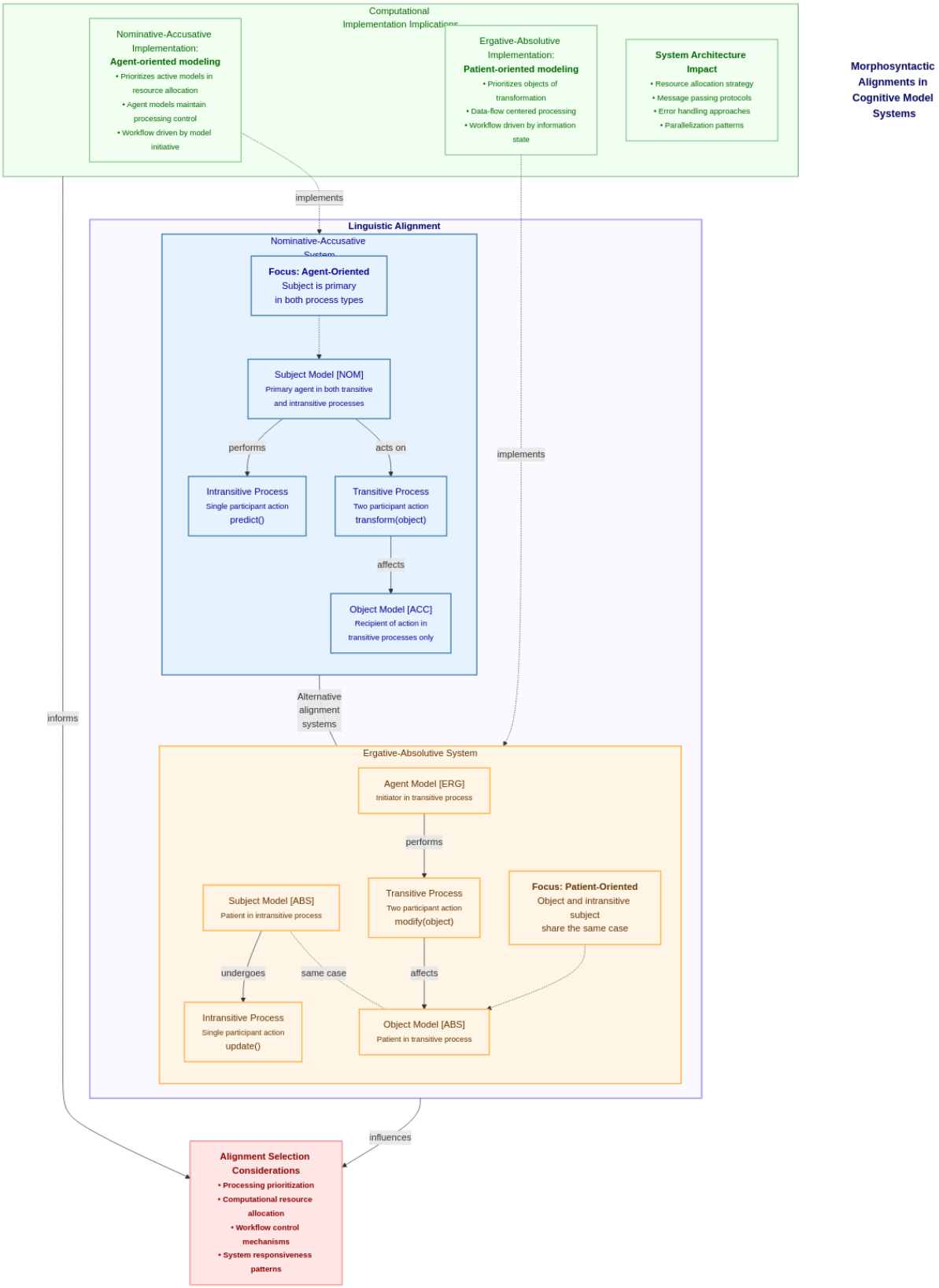


Figure 6: Figure 6. Morphosyntactic Alignments in Model Relationships. This diagram illustrates two fundamental alignment patterns that can be applied to model relationships in CEREBRUM, derived from linguistic morphosyntactic structures. The Nominative-Accusative alignment (left) groups subject models 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

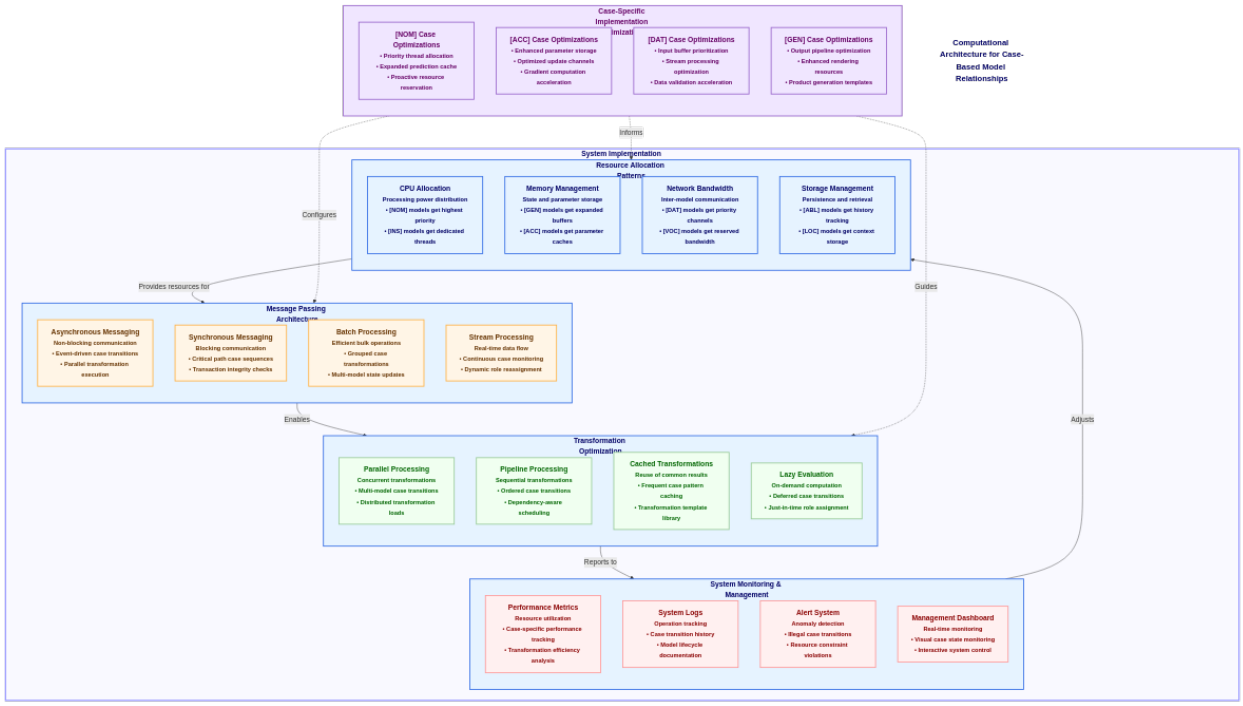


Figure 7: Figure 7. Computational Implementation of Model Relationships. This diagram illustrates the practical computational architecture required to implement CEREBRUM's case-based model relationships in real-world systems. The implementation encompasses four interconnected components that translate 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.

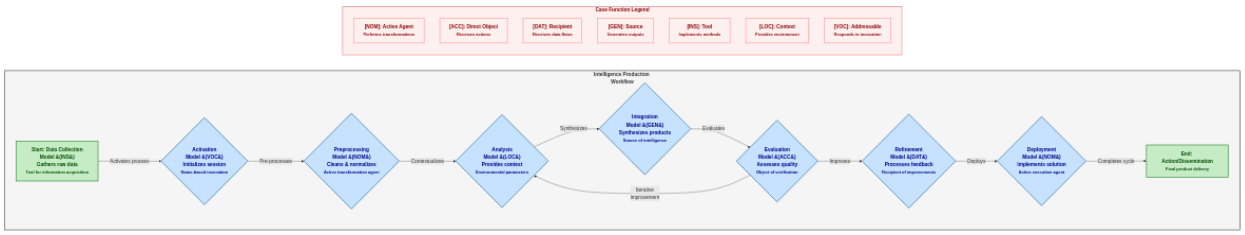


Figure 8: Figure 8. Intelligence Production Workflow with Case-Bearing Models. This flowchart depicts the intelligence production cycle from data collection to dissemination, highlighting how models in different case roles support specific stages of the workflow. The process begins with data collection 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.

This cyclical process demonstrates how case transformations enable models within the CEREBRUM framework to maintain their core identity while dynamically adapting to different functional requirements throughout the intelligence production lifecycle. Each case assignment optimizes specific aspects of model behavior—from data collection and processing to product generation and quality assessment—creating a flexible yet rigorously structured approach to managing intelligence workflows.

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.

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.

Cross-Domain Integration Benefits

The CEREBRUM framework'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

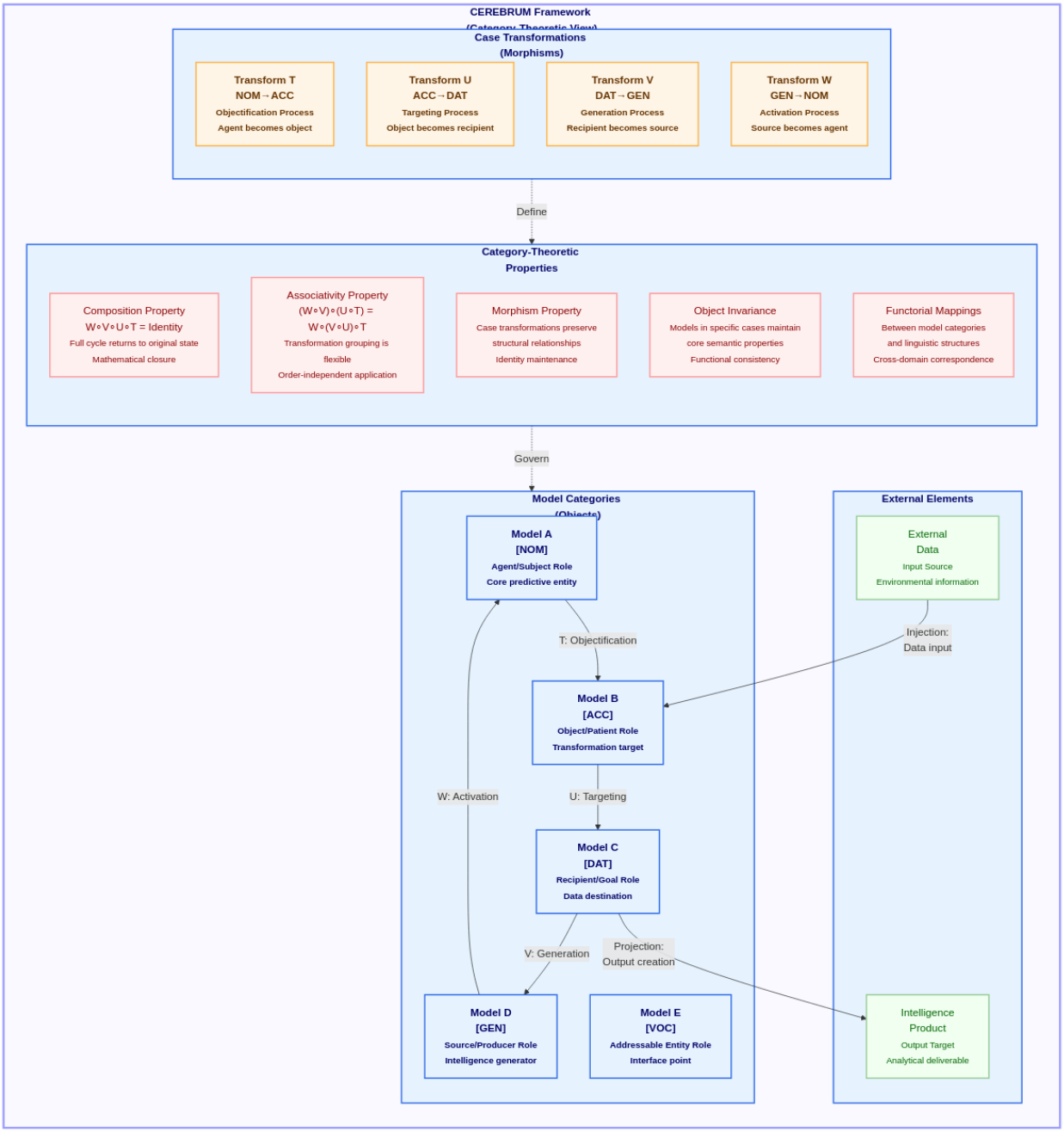


Figure 9: Figure 9. CEREBRUM Category Theory Framework. This diagram formalizes the CEREBRUM framework using category theory, providing a rigorous mathematical foundation for model transformations. The framework represents cognitive models as objects in a category, with different case assignments (Nominative, 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.

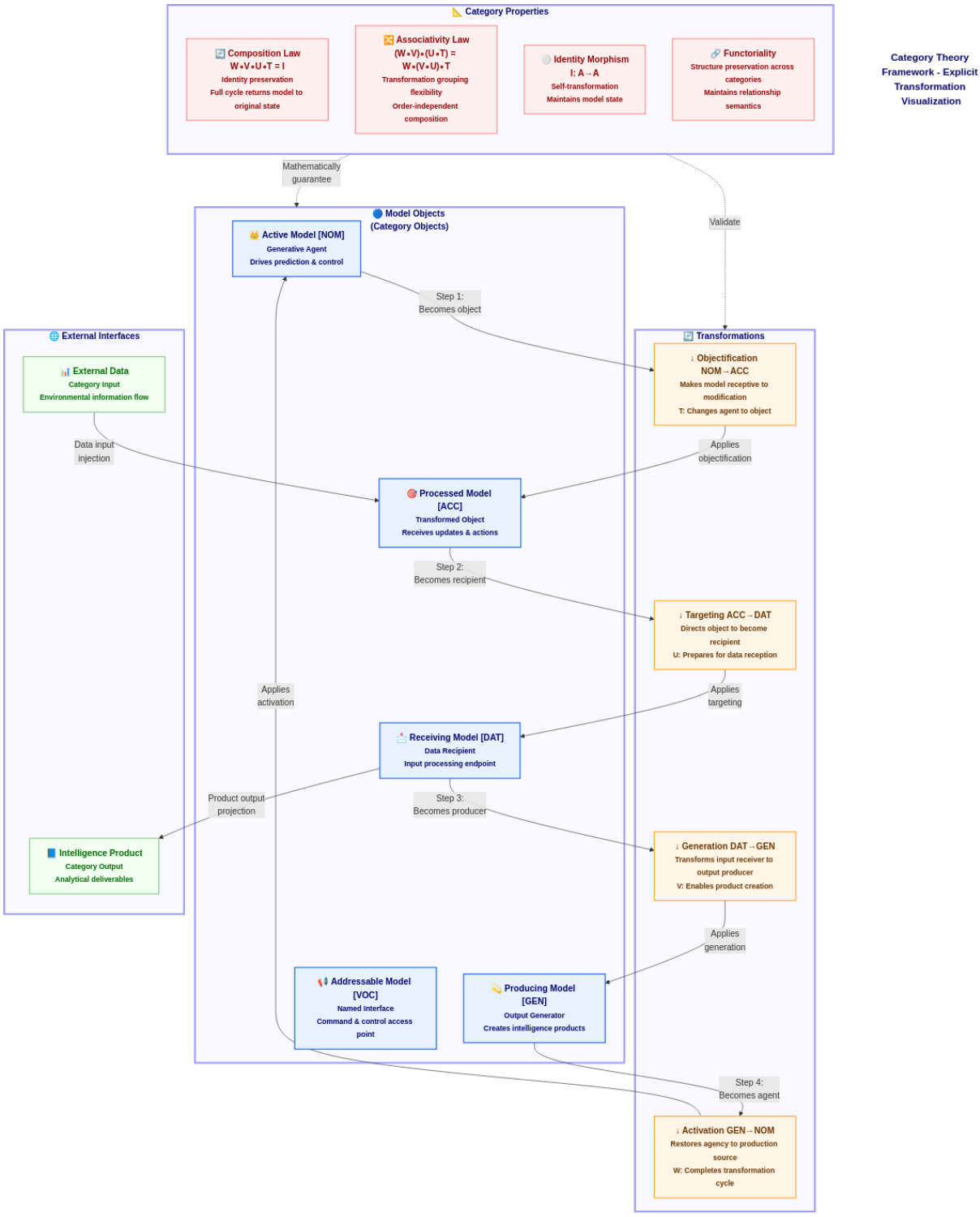


Figure 10: Figure 10. Category Theory Framework - Alternative Visualization. This diagram provides a complementary perspective to Figure 7, offering a more explicit representation of the transformations between case-bearing models in CEREBRUM. While Figure 7 presents the category-theoretic structure 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),

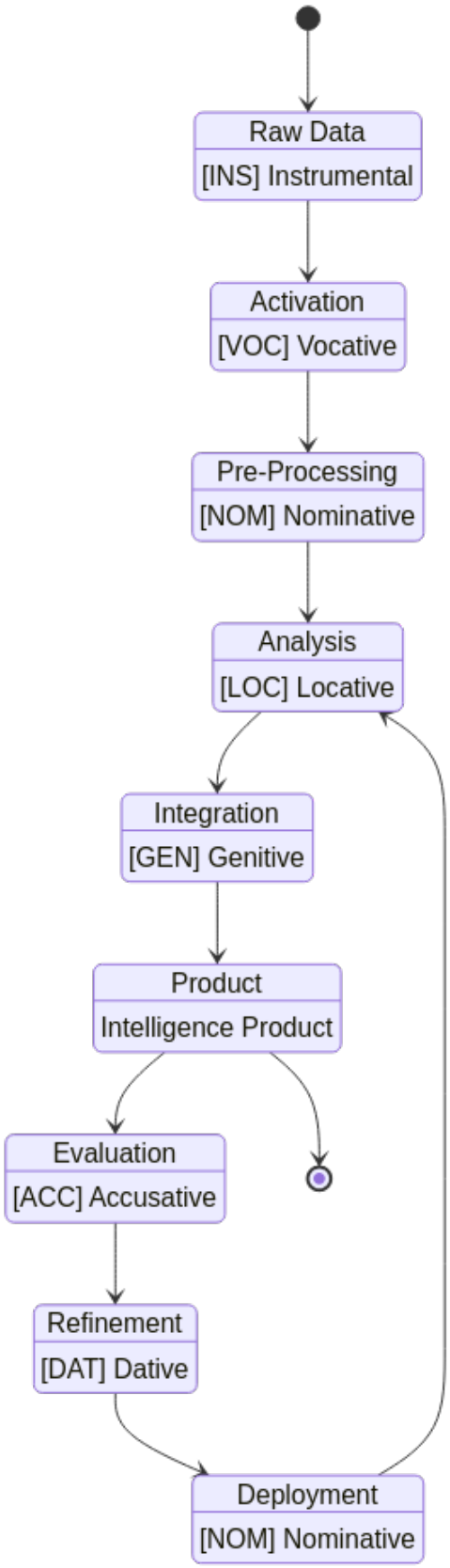


Figure 11: Figure 11. Intelligence Production Workflow State Diagram. This state diagram visualizes the transformation of data into intelligence products through a series of interconnected states. Beginning with raw data collection (Instrumental case [INS]), the process moves through system activation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

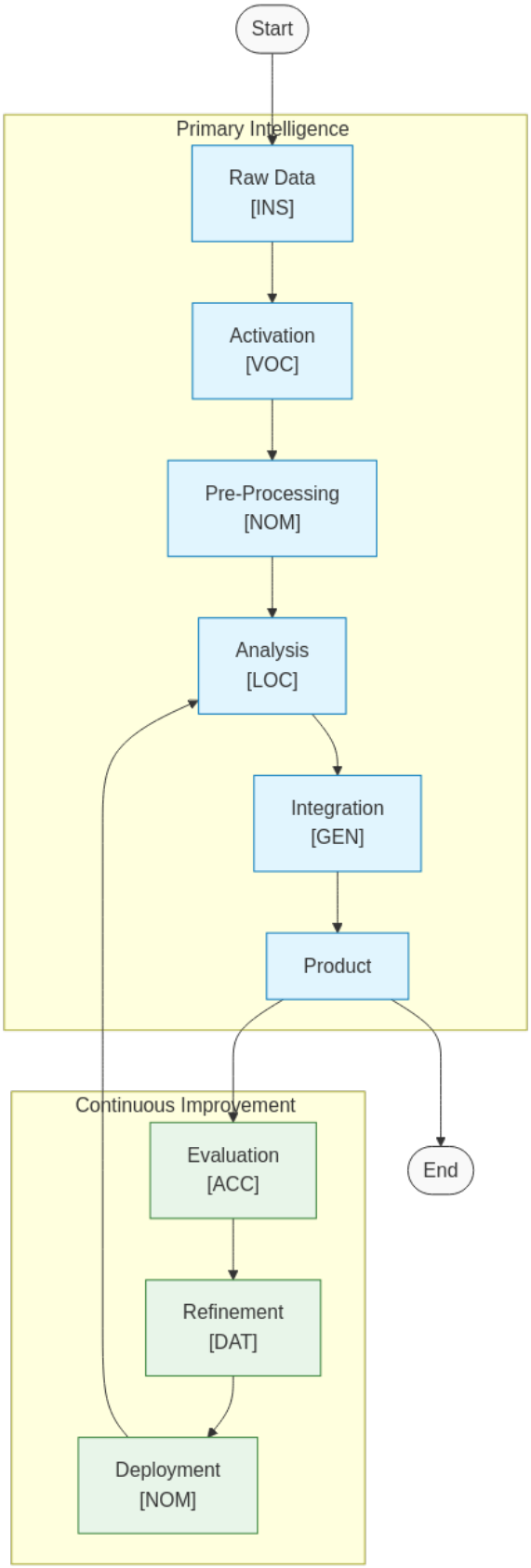


Figure 12: Figure 12. Alternative Visualization of Intelligence Production Workflow. This diagram presents a complementary perspective to Figure 11, organizing the intelligence workflow into two distinct components: the main Intelligence Workflow and a separate Feedback Loop. The main workflow proceeds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

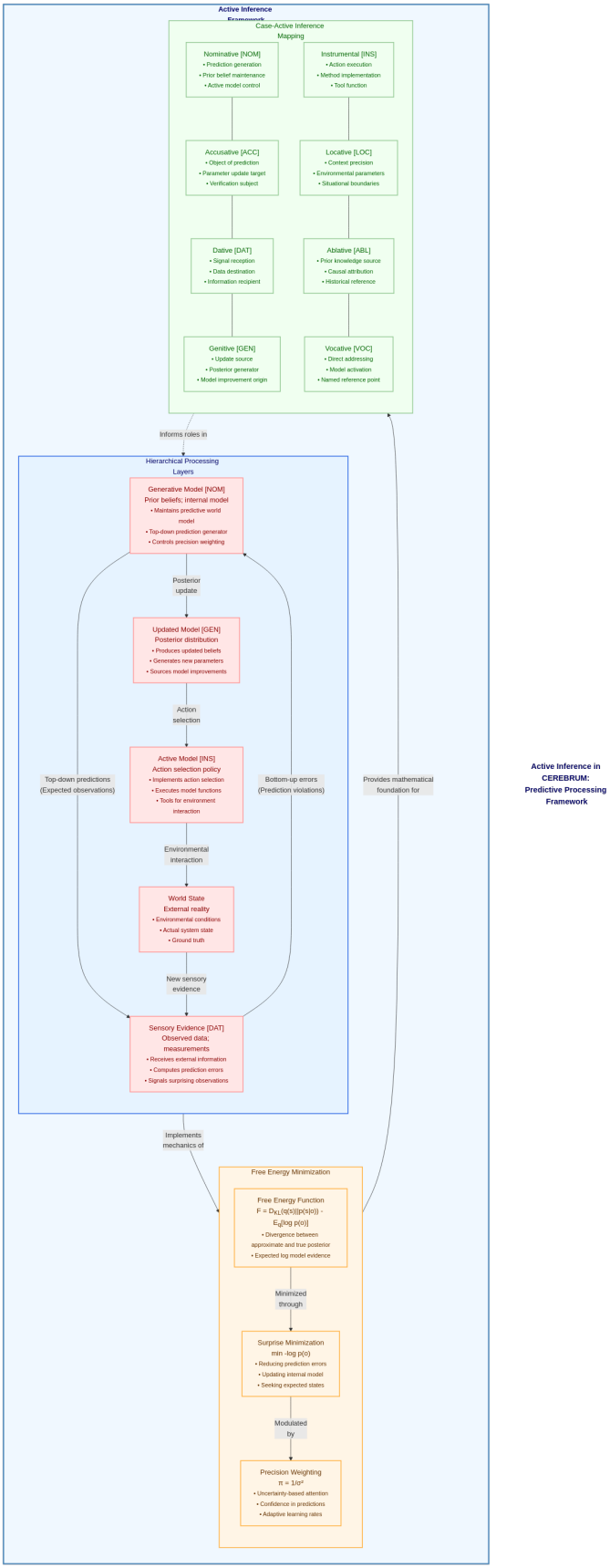


Figure 13: Figure 13. Active Inference Integration Framework. This diagram illustrates how CEREBRUM integrates with active inference principles, showing the mapping between case assignments and predictive processing mechanisms. The framework is organized into three interconnected components. The Processing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

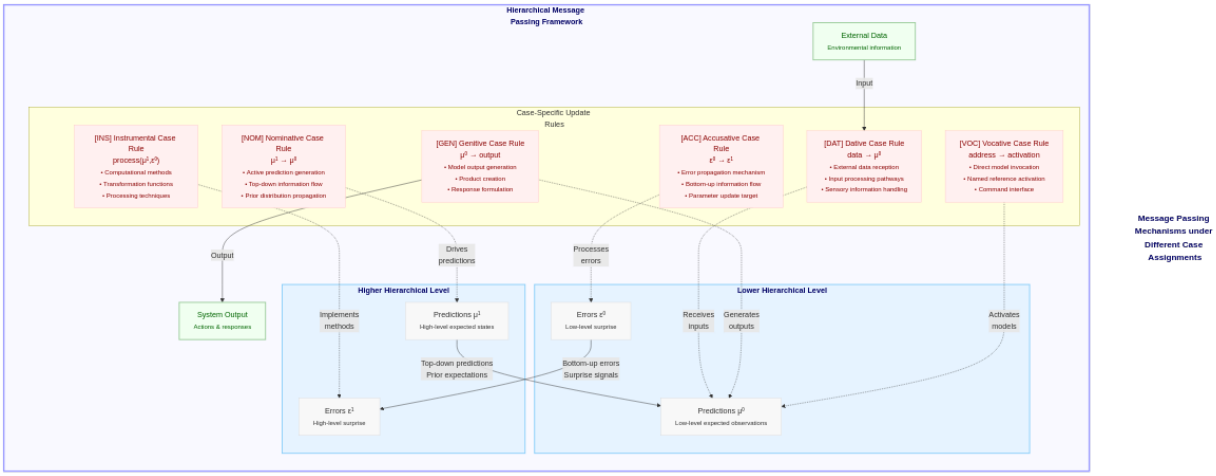


Figure 14: Figure 14. Case-Specific Message Passing in Active Inference. This diagram details the specific message passing mechanisms that implement case transformations within CEREBRUM's active inference framework. The central component shows the hierarchical bidirectional message passing that characterizes 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.

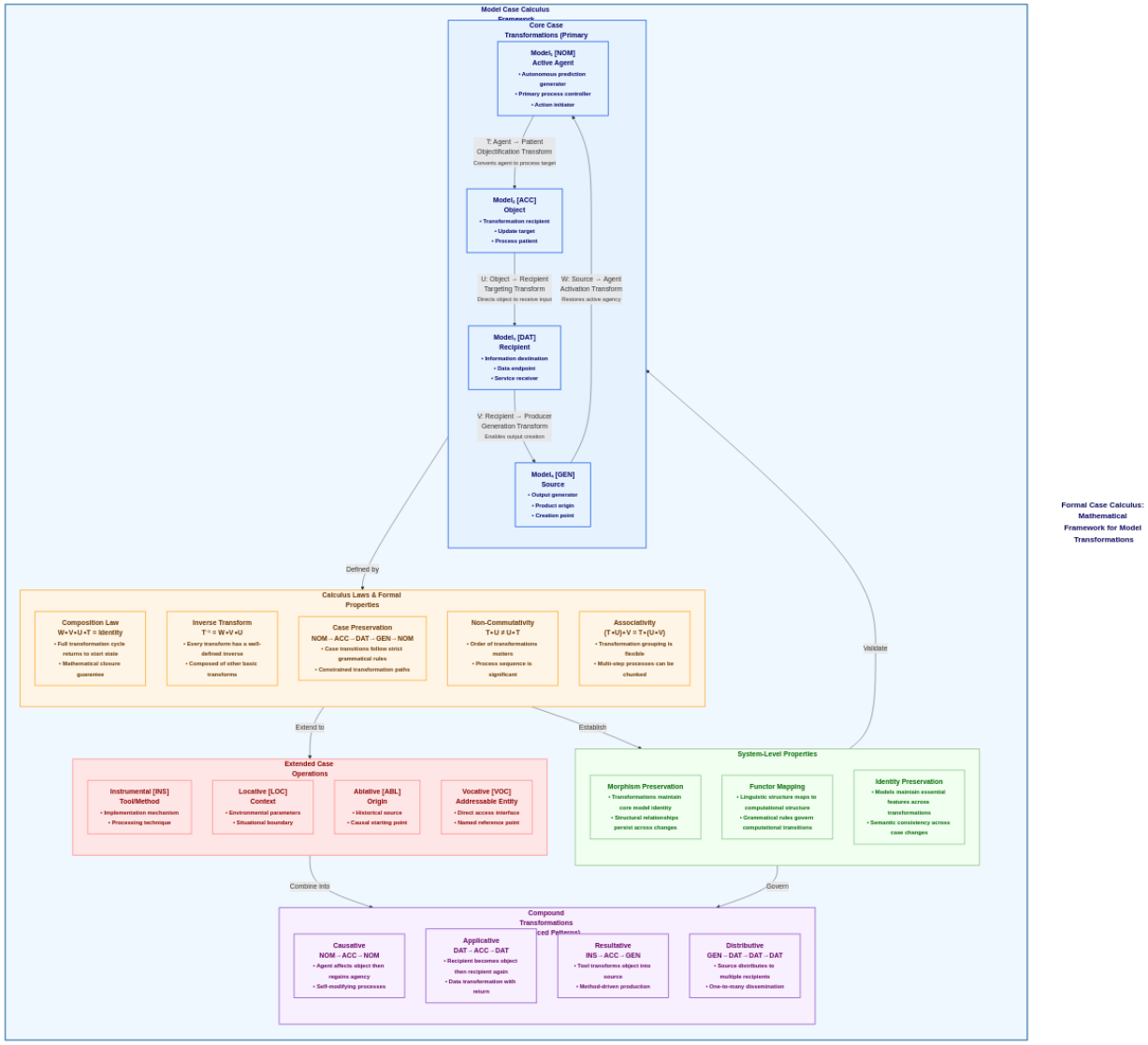


Figure 15: Figure 15. Model Case Calculus Framework. This diagram presents the formal mathematical calculus underlying CEREBRUM's case transformations, providing a rigorous foundation for modeling dynamic role changes in cognitive systems. The Core Case Transformations section shows the primary cycle 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	Systematic relationship framework; grammatical role templates; morphosyntactic structures	Structured representation of model interactions; formalized functional transitions; systematic role assignment	Provides formal semantics for model relationships; enables compositional theory of model interactions
Cognitive Systems Modeling	Entity representation and processing; model formalization; information-processing structures	Flexible model instantiation across functional roles; adaptive model morphology; unified modeling paradigm	Advances theory of cognitive model composition; formalizes functional transitions in cognitive systems
Active Inference	Predictive transformation mechanics; free energy principles; precision-weighted learning	Self-optimizing workflows with error minimization; principled uncertainty handling; bidirectional message passing	Extends active inference to model ecosystems; provides mathematical foundation for case transformations
Intelligence Production	Practical operational context; analytical workflows; intelligence cycle formalisms	Real-world application in case management systems; operational coherence; analytical integrity	Bridges theoretical and applied intelligence; enhances intelligence workflow coherence; improves product quality

Related Work

CEREBRUM synthesizes concepts from several established research traditions. While this initial paper focuses on presenting the core framework without extensive citations, 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. Unlike traditional architectures where components often have fixed functions, CEREBRUM allows 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. This provides a principled foundation for coordinating complex model ecosystems.

Category-Theoretic Cognition

The formalization of CEREBRUM using category theory provides a basis for compositional reasoning about cognitive systems. Representing case transformations as functors enables formal verification of properties like identity preservation across functional transitions. This supports reasoning about model composition, transformation sequencing, and structural relationships, aligning with the compositional nature of cognition.

Active Inference Applications

CEREBRUM extends active inference from individual processes to entire model ecosystems. It treats case transformations as precision-weighted processes minimizing free energy across the system. This enables principled coordination by explicitly representing functional relationships through case assignments and formalizing interfaces with precision-weighting, creating intelligent workflows where models cooperate to minimize system-wide free energy.

Linguistic Computing

CEREBRUM exemplifies a linguistic computing approach, applying declensional semantics to model management. It treats models as entities assuming different morphological forms based on functional roles, mirroring noun declension in natural languages. Implementing computable declension paradigms allows for more expressive and flexible representations of model relationships than traditional paradigms, bridging natural and artificial intelligence systems.

(See Supplement 2: Novel Linguistic Cases for a discussion on discovering and creating new linguistic cases.) (See Supplement 3: Practical Applications for detailed implementation examples.) (See Supplement 7: Computational Complexity for an analysis of resource scaling characteristics across different case assignments.)

Conclusion

CEREBRUM provides a structured framework for managing cognitive models by applying linguistic case principles to represent different functional roles and relationships. This synthesis of linguistic theory, category mathematics, active inference, and intelligence production creates a powerful paradigm for understanding and managing complex model ecosystems. By treating models as case-bearing entities, CEREBRUM enables more formalized transformations between model states while providing intuitive metaphors for model relationships that align with human cognitive patterns and operational intelligence workflows.

The formal integration of variational free energy principles with case transformations 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.

The CEREBRUM framework represents another milestone in a long journey of how we conceptualize model relationships, moving from ad hoc integration approaches, on through seeking the first principles of persistent, composable, linguistic intelligences. This journey, really an adventure, continues to have profound implications for theory and practice. By here incipiently formalizing the grammatical structure of model interactions, CEREBRUM points towards enhancement of 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 yet flexible approaches to model management will become increasingly essential for maintaining conceptual coherence and operational effectiveness.

Supplement 1: Mathematical Formalization

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

1.1 Variational Free Energy and Case Transformations

Equation 1: Variational Free Energy for Case Transformation

$$F = D_{KL}[q(s|T(m))||p(s|m)] - \mathbb{E}_p[\log p(o|s, T(m))] \tag{1}$$

where T(m) represents the transformed model, s are internal states, and o are observations.

Equation 2: Markov Blanket and Case Relationship

$$\text{Case}(M) \subseteq \text{MB}(M) \tag{2}$$

where MB(M) denotes the Markov blanket of model M.

Equation 3: Precision Weighting for Case Selection

$$\beta(c, m) = \frac{\exp(-F(c, m))}{\sum_i \exp(-F(c_i, m))} \tag{3}$$

where (c,m) is the precision weight for case c and model m.

Equation 4: Case-Specific Gradient Descent on Free Energy

$$\frac{\partial m}{\partial t} = -\kappa_c \cdot \frac{\partial F}{\partial m} \tag{4}$$

where κ_c is the case-specific learning rate.

Equation 5: Expected Free Energy Reduction in Case Transitions

$$\mathbb{E}[\Delta F] = \sum_{s,a} T(s'|s, a)\pi[a|s](F(s, c) - F(s', c')) \tag{5}$$

where c and c' represent the initial and target cases respectively.

Equation 6: Bayes Factor for Case Selection

$$BF = \frac{p(o|m, c_1)}{p(o|m, c_2)} \tag{6}$$

Equation 7: Free Energy Minimization in Case Transitions

$$F = D_{KL}[q(s|c, m)||p(s|m)] - \mathbb{E}_{q(s|c,m)}[\log p(o|s, c, m)] \tag{7}$$

1.2 Message Passing Rules for Different Cases

These equations illustrate how case assignments modulate standard hierarchical message passing (e.g., in predictive coding) where beliefs/predictions (μ) and prediction errors (ε) flow between adjacent levels (denoted by superscripts 0 and 1). The case-specific weights (κ_c) determine the influence of each message type based on the model's current functional role.

Equations 8-12: Case-Specific Message Passing Rules

$$\text{Nominative [NOM]} : \mu^0 = \mu^0 + \kappa_{NOM} \cdot (\mu^1 - \mu^0) \tag{8}$$

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

$$\text{Accusative [ACC]} : \varepsilon^1 = \varepsilon^1 + \kappa_{ACC} \cdot (\varepsilon^0 - \varepsilon^1) \tag{9}$$

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

$$\text{Dative [DAT]} : \mu^0 = \mu^0 + \kappa_{DAT} \cdot (data - \mu^0) \tag{10}$$

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

$$\text{Genitive [GEN]} : output = \mu^0 + \kappa_{GEN} \cdot \eta \tag{11}$$

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

$$\text{Instrumental [INS]} : process = f(\mu^1, \varepsilon^0) \cdot \kappa_{INS} \tag{12}$$

(A process output determined by some function f of top-down prediction μ^1 and bottom-up error ε^0, weighted by κ_{INS})

$$\text{Vocative [VOC]} : activation = \sigma(\kappa_{VOC} \cdot sim(id, address)) \tag{12a}$$

(Activation state determined by similarity between model identity id and incoming address, weighted by κ_{VOC} and passed through activation function σ)

where κ_c represents case-specific learning rates or precision weights, η is a noise term, μ^0, μ^1 represent beliefs/predictions, and $\varepsilon^0, \varepsilon^1$ represent prediction errors at adjacent hierarchical levels.

1.3 Precision Allocation and Resource Optimization

Equation 13: Precision Weight Allocation with Temperature

$$\beta(c, m) = \frac{\exp(-\gamma \cdot F(c, m))}{\sum_i \exp(-\gamma \cdot F(c_i, m))} \tag{13}$$

where is the inverse temperature parameter controlling allocation sharpness.

Equation 14: Resource-Weighted Free Energy

$$F_\beta(m) = \sum_c \beta(c, m) \cdot F(c, m) \cdot R(c) \tag{14}$$

where R(c) represents the computational resources allocated to case c.

1.4 Novel Case Formalizations

Equation 15: Conjunctive Case Free Energy

$$F_{CNJ} = D_{KL}[q(s|CNJ, m)||p(s|m)] - \mathbb{E}_{q(s|CNJ,m)}[\log p(o|s, \{m_i\})] \tag{15}$$

where {m_i} represents the assembly of connected models.

Equation 16: Conjunctive Case Message Passing

$$\mu^{CNJ} = \sum_i w_i \cdot \mu_i + \kappa_{CNJ} \cdot (\prod_i \mu_i - \sum_i w_i \cdot \mu_i) \tag{16}$$

where w_i are model-specific weighting factors.

Equation 17: Recursive Case Precision Dynamics

$$\beta(REC, m) = \frac{\exp(-\gamma \cdot F(REC, m))}{\sum_i \exp(-\gamma \cdot F(c_i, m)) + \exp(-\gamma \cdot F(REC, m))} \tag{17}$$

1.5 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.

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 Speculative Novel Case: The Emergent "Conjunctive" Case

One speculative example of a novel case that might emerge within CEREBRUM is what we might term the "conjunctive" case [CNJ]. This case would represent a model's role in synthesizing multiple predictive streams into coherent joint predictions that couldn't be achieved through simple composition of existing cases.

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.

2.4 Speculative Novel Case: The “Recursive” Case

Another potential novel case is the “recursive” case [REC], which would enable a model to apply its transformations to itself, creating a form of computational reflection not captured by traditional cases.

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.

2.5 Speculative Novel Case: The “Metaphorical” Case

A third potential novel case is the “metaphorical” case [MET], which would enable a model to map structures and relationships from one domain to another, creating computational analogies that transfer knowledge across conceptual spaces.

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.

2.6 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 languagedescribing 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 cognitionas 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.

Table A1: Properties of Speculative Novel Cases in CEREBRUM

Property	Conjunctive Case [CNJ]	Recursive Case [REC]	Metaphorical Case [MET]
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
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
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
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

Property	Conjunctive Case [CNJ]	Recursive Case [REC]	Metaphorical Case [MET]
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

Supplement 3: Practical Applications

The declension paradigm for cognitive models offers significant practical benefits in complex model ecosystems. This supplement provides concise examples, heuristics, and implementation recipes for applying case declensions in real-world systems.

3.1 Model Pipeline Optimization

Complex cognitive workflows typically involve sequences of models arranged in processing pipelines. By applying case declensions, each component can seamlessly adapt its interfaces.

3.2 Implementation Recipe: Pipeline Adapter Pattern

```python
def transform_model(model, target_case):
    """Transform model to target case with appropriate configuration"""
    case_configs = {
        "NOM": {"input_gates": False, "output_gates": True, "precision": 0.9},
        "ACC": {"input_gates": True, "output_gates": False, "precision": 0.8},
        "DAT": {"input_gates": True, "output_gates": True, "precision": 0.7},
        "GEN": {"input_gates": False, "output_gates": True, "precision": 0.9}
    }

    # Apply case configuration to model
    for param, value in case_configs.get(target_case, {}).items():
        setattr(model, param, value)

    return model

def optimize_pipeline(models):
    """Assign optimal cases to models in pipeline"""
    if not models: return []

    # Assign cases based on position in pipeline
    models[0].case = "NOM"  # First model generates

    for i in range(1, len(models)-1):
        models[i].case = "DAT"  # Middle models forward

    if len(models) > 1:
        models[-1].case = "GEN"  # Last model produces

    return models
```

3.3 Pipeline Design Patterns

Pattern	Case Sequence	Use Case	Key Benefit
Linear	NOMDATGEN	Sequential processing	Simple, efficient data flow
Branching	NOM(DAT,DAT)GEN	Parallel processing	Increased throughput
Aggregating	(NOM,NOM)ACCGEN	Multi-source fusion	Information integration
Feedback	NOMDATGENLOCNOM	Iterative refinement	Self-correction

3.4 Resource Allocation Strategies

Table: Optimized Resource Allocation by Task Type

Task Type	Priority Case Order	Resource Distribution	Optimization Goal
Real-time decision	NOM > DAT > ACC	50% generation, 30% routing, 20% processing	Minimize latency
Data processing	ACC > DAT > GEN	50% processing, 30% routing, 20% output	Maximize throughput
Report generation	GEN > NOM > LOC	50% output, 30% content, 20% context	Optimize clarity
Method development	INS > ACC > NOM	50% method, 30% testing, 20% generation	Minimize errors

3.5 Implementation Recipe: Resource Allocator

```python
def allocate_resources(models, task_type, total_compute):
    """Allocate computational resources based on task priorities"""
    # Priority maps by task
    priorities = {
        "real_time_decision": {"NOM": 0.5, "DAT": 0.3, "ACC": 0.2},
        "data_processing": {"ACC": 0.5, "DAT": 0.3, "GEN": 0.2},
        "report_generation": {"GEN": 0.5, "NOM": 0.3, "LOC": 0.2}
    }

    # Get priorities for this task
    case_weights = priorities.get(task_type, {"NOM": 0.25, "ACC": 0.25, "DAT": 0.25, "GEN": 0.2

    # Group models by case and allocate resources
    case_groups = {}
    for model in models:
        if model.case not in case_groups:
            case_groups[model.case] = []
        case_groups[model.case].append(model)

    # Apply allocations
    for model in models:
        weight = case_weights.get(model.case, 0.1)
        count = len(case_groups.get(model.case, [1]))
        model.compute_allocation = (weight * total_compute) / count

    return models
```

3.6 Resource Allocation Heuristics

1. **Dynamic Scaling Rules:**
   - Scale up [NOM] case precision during initial processing
   - Reduce [ACC] case precision under resource constraints
   - Balance [DAT] case resources based on pipeline depth
   - Prioritize [GEN] case for final output quality
2. **Practical Load Balancing:**
   - Distribute 50% resources to primary case functions
   - Allocate 30% to auxiliary processing

- Reserve 20% for error handling and recovery
- Adjust allocations dynamically based on performance metrics

3.7 Model Ecosystem Adaptability

Table: Context-Specific Adaptation Patterns

Context Change	Case Transition	Implementation Approach	Expected Outcome
Data volume spike	NOMACC	Increase buffer capacity, batch processing	Sustained throughput
Accuracy requirement	ACCNOM	Precision increase, additional validation	Higher quality results
Latency constraints	INSNOM	Pipeline shortening, parallelization	Faster response time
Novel data	ACCABL	Representation adaptation, uncertainty handling	Better generalization
Resource limitation	AllACC selective	Selective processing, prioritization	Resource conservation

3.8 Implementation Recipe: Adaptive Configuration

```
def reconfigure_ecosystem(models, context):
    """Reconfigure model ecosystem for different operational contexts"""
    # Configuration templates for common contexts
    configurations = {
        "high_throughput": {
            "processors": ["ACC", "DAT", "DAT"],
            "reasoners": ["NOM", "INS"],
            "outputs": ["GEN"]
        },
        "high_accuracy": {
            "processors": ["NOM", "ACC"],
            "reasoners": ["INS", "LOC", "INS"],
            "outputs": ["NOM", "GEN"]
        },
        "low_latency": {
            "processors": ["NOM", "DAT"],
            "reasoners": ["NOM"],
            "outputs": ["GEN"]
        }
    }

    # Apply configuration if available
    config = configurations.get(context)
    if not config:
        return False

    # Group and assign cases
    for model in models:
        if model.type in config:
            cases = config[model.type]
            if cases:
```

```
                model.case = cases[0]
                cases.append(cases.pop(0))  # Rotate for balanced assignment

    return True
```

3.9 Knowledge Graph Enhancement

3.10 Implementation Recipe: Case-Based Knowledge Graph

```
def build_knowledge_graph(entities, relationships):
    """Build knowledge graph with case-semantic relationships"""
    # Map cases to semantic relations
    case_relations = {
        "NOM": "produces", "ACC": "targets",
        "DAT": "transfers", "GEN": "sources",
        "INS": "implements", "LOC": "contextualizes",
        "ABL": "originates", "VOC": "communicates"
    }

    # Build graph with semantic edges
    graph = {}
    for source, case, target in relationships:
        relation = case_relations.get(case, "relates_to")

        if source not in graph:
            graph[source] = []

        graph[source].append({
            "target": target,
            "relation": relation,
            "case": case
        })

    return graph
```

Table: Case-Based Knowledge Representation

Case	Relation Type	Graph Properties	Query Pattern	Example Application
NOM	Generation	Outward, weighted	`source --[produces]-->?`	Find model outputs
ACC	Reception	Inward, typed	`?--[targets]->target`	Find data consumers
DAT	Transfer	Bidirectional	`node--[transfers]-->?`	Trace information flow
GEN	Source	Outward, authentic	`?--[sources]->target`	Find authoritative sources
INS	Method	Procedural	`process--[implements]->?`	Find implementations

Case	Relation Type	Graph Properties	Query Pattern	Example Application
LOC	Context	Situational	`event--[contextualizes]->?`	Find relevant context

3.11 Practical Implementation Patterns

Table: Implementation Patterns by System Type

System Type	Key Cases	Implementation Strategy	Success Metrics
LLM reasoning	INS, LOC, GEN	Consistent prompt-output interfaces	Reasoning accuracy, explanation quality
Multimodal	NOM, ACC, LOC	Cross-modal alignment in shared space	Cross-modal inference performance
Autonomous agents	NOM, DAT, GEN	Balance reactivity with planning	Goal achievement rate, adaptation speed
Federated learning	ACC, ABL, VOC	Privacy-preserving knowledge exchange	Learning efficiency with privacy
Recommendation	GEN, ACC, DAT	Personalization through case precision	Relevance, diversity, novelty

3.12 Quick Reference: Case Selection Decision Tree

```
1. Model's PRIMARY ROLE:
    GENERATES content  NOM
    RECEIVES data  ACC
    TRANSFERS information  DAT
    PRODUCES output  GEN
    PROVIDES methods/context  INS/LOC

2. POSITION in pipeline:
    FIRST component  NOM
    MIDDLE component  DAT
    JUNCTION component  DAT+ACC
    FINAL component  GEN

3. SPECIAL FUNCTION:
    ERROR handling  INS+LOC
    MEMORY systems  ABL+GEN
    INTERACTIVE systems  VOC+NOM
    LEARNING components  ACC+NOM
```

3.13 Implementation Best Practices

1. **Development Workflow:**
    - Begin with core cases (NOM, ACC, GEN) for basic functionality
    - Add specialized cases only when needed for specific capabilities
    - Test case transitions under varying load conditions
    - Monitor case-specific performance metrics

2. **Optimization Strategies:**
    - Profile each case's resource usage separately
    - Identify and address bottlenecks in case transitions
    - Apply case-specific precision scaling under resource constraints
    - Consider hardware acceleration for dominant cases
3. **Debugging Approach:**
    - Trace information flow through case transitions
    - Verify interface compatibility between connected cases
    - Check resource allocation balance across cases
    - Test graceful degradation through case simplification

This appendix provides practical guidance for implementing the CEREBRUM framework in real-world cognitive systems. By following these recipes, heuristics, and patterns, developers can leverage case declension to create more flexible, efficient, and robust model ecosystems.

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 (Coecke et al., 2010): - Uses monoidal categories to formalize compositional meaning in natural language - Implements tensor product representations of linguistic structures - Provides mathematical foundations for semantic composition - Limitations: Focuses primarily on linguistic meaning rather than broader cognitive processes

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 to case relationships and transformations. By formalizing case functors, natural transformations, and commutative diagrams for model interactions, CEREBRUM provides 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.

4.3.2 Morphological Computing

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 to model management, treating models as entities that can assume different morphological forms based on their functional roles. 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 require-

ments 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.

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.

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.

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:

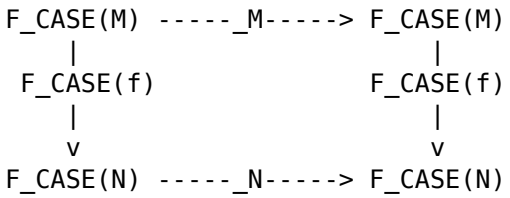
$$\eta : F_{\text{CASE}_1} \Rightarrow F_{\text{CASE}_2}$$

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

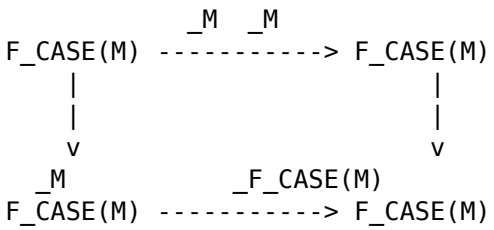
For any model M and two cases CASE_1 and CASE_2, the following diagram commutes:



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

This supplement expands on the future directions briefly outlined in the main text, providing a detailed roadmap for theoretical and practical developments of the CEREBRUM framework. Each direction is accompanied by technical challenges, potential implementation approaches, and expected outcomes.

6.1 Programming Libraries and Implementation Frameworks

6.1.1 Technical Challenges

The development of robust programming libraries for CEREBRUM faces several technical challenges:

1. **Polymorphic Type Systems**: Implementing case-bearing models requires sophisticated type systems that can represent morphological transformations while maintaining type safety.
2. **Interface Adaptability**: Dynamic reconfiguration of model interfaces based on case assignments requires flexible interface definitions and runtime adaptation mechanisms.
3. **Performance Optimization**: Case transformations must be efficiently implemented to minimize computational overhead, particularly for real-time applications.
4. **Cross-Language Compatibility**: CEREBRUM implementations should maintain consistent semantics across different programming languages and paradigms.

6.1.2 Proposed Implementation Approaches

1. **Core Language-Agnostic Specification**:

```
# Pseudocode for CEREBRUM model interface
class CaseModel<T>:
  enum Case { NOM, ACC, DAT, GEN, INS, LOC, ABL, VOC }

  # Core model properties
  P: ParametricStructure
  S: StateSpace<T>
  : Parameters
  I: InputInterfaces
  O: OutputInterfaces
  C: Case

  # Case transformation method
  transform(targetCase: Case): CaseModel<T>

  # Free energy calculation
  calculateFreeEnergy(): Real
```

2. **Language-Specific Implementations**:
   - Functional implementations (Haskell, OCaml) focusing on type-safe transformations
   - Object-oriented implementations (Python, Java) emphasizing inheritance and polymorphism
   - Low-level implementations (C++, Rust) prioritizing performance and memory efficiency

6.1.3 Evaluation Metrics

Progress in library development will be measured by: 1. API completeness and consistency with the theoretical framework 2. Performance benchmarks across different case transformations 3. Integration capabilities with existing cognitive modeling frameworks 4. User adoption and community contributions

6.2 Visualization Tools for Case Transformations

6.2.1 Technical Challenges

Creating effective visualization tools for CEREBRUM presents unique challenges:

1. **Multi-Dimensional Representation**: Case transformations involve changes across multiple dimensions (parameters, interfaces, precision weights) that must be visually represented.
2. **Temporal Dynamics**: Visualizing the dynamics of case transformations over time requires sophisticated animation and transition effects.
3. **Hierarchical Visualization**: Representing nested model relationships and transformations requires hierarchical visualization techniques.
4. **Interactive Exploration**: Enabling users to interactively explore and manipulate case transformations requires responsive interfaces and real-time computation.

6.2.2 Proposed Visualization Approaches

1. **Morphological Transformation Maps**:
   - Interactive diagrams showing parameter and interface changes during transformations
   - Heat maps representing precision weighting shifts across case transformations
   - Animated transitions between case states with interpolated intermediate states
2. **Case Relationship Networks**:
   - Graph-based visualizations of model ecosystems with edges representing transformation relationships
   - Force-directed layouts adapting to dynamic changes in model relationships
   - Visual encodings of information flow between models in different cases
3. **Work Process Visualization**:
   - Timeline-based views of intelligence workflows with case-specific activities
   - Sankey diagrams showing resource allocation across models in different cases
   - Decision tree visualizations for case selection processes

6.2.3 Evaluation Criteria

Visualization tools will be evaluated based on: 1. Clarity of representation and information density 2. Interactive responsiveness and exploratory capabilities 3. Integration with computational implementations 4. Effectiveness in communicating complex transformations to users

6.3 Linguistic Extensions Beyond Case Systems

6.3.1 Technical Challenges

Extending CEREBRUM to incorporate additional linguistic features presents several challenges:

1. **Formal Integration**: Integrating features like aspect, tense, and modality requires formal definitions that align with the existing case framework.

2. **Compositional Semantics**: Ensuring that combinations of linguistic features (e.g., case + aspect) have well-defined semantics and transformations.
3. **Cross-Linguistic Validation**: Validating that the extended framework remains applicable across diverse linguistic patterns.
4. **Computational Complexity**: Managing increased complexity from additional linguistic dimensions without exponential growth in computational requirements.

6.3.2 Proposed Extensions

1. **Aspectual Framework**:
   - **Perfective Aspect**: Models optimized for completed processes with emphasis on outcomes
   - **Imperfective Aspect**: Models focused on ongoing processes with temporal dynamics
   - **Iterative Aspect**: Models specialized for repeated operations with cycle optimization
2. **Temporal Framework**:
   - **Past Tense**: Models representing historical states and causal precedence
   - **Present Tense**: Models operating in real-time with synchronous processing
   - **Future Tense**: Models projecting forward states with predictive emphasis
3. **Modal Framework**:
   - **Alethic Modality**: Models representing physical necessity and possibility
   - **Epistemic Modality**: Models encoding certainty and knowledge states
   - **Deontic Modality**: Models incorporating normative constraints and permissions

6.3.3 Research Methodology

The exploration of linguistic extensions will follow a structured approach: 1. Formal definition of extensions with category-theoretic foundations 2. Computational implementation of prototype extensions 3. Empirical validation through case studies in intelligence production 4. Documentation and integration into the core CEREBRUM framework

6.4 Open Source Community Development

6.4.1 Governance Challenges

Establishing effective open source governance for CEREBRUM involves:

1. **Community Engagement**: Attracting and retaining contributors from diverse backgrounds and expertise levels.
2. **Quality Assurance**: Maintaining theoretical consistency and implementation quality across contributions.
3. **Version Management**: Handling library versioning and backward compatibility across the ecosystem.
4. **Knowledge Transfer**: Ensuring effective documentation and onboarding for new contributors.

6.4.2 Proposed Governance Structure

1. **Technical Steering Committee** (TSC):
   - Responsible for framework architecture and core specifications
   - Reviews and approves significant architectural changes
   - Ensures theoretical consistency across implementations
2. **Working Groups**:

- Language-specific implementation groups
- Theoretical development group
- Application domain groups (e.g., intelligence production, cognitive science)
- Documentation and education group

3. **Community Processes**:
   - Regular community calls for synchronization
   - Transparent decision-making through RFC processes
   - Mentorship programs for new contributors
   - Regular hackathons and workshops for collaborative development

6.4.3 Success Metrics

Community development will be evaluated based on: 1. Number and diversity of active contributors 2. Quality and quantity of contributions across different areas 3. Adoption in academic and industry settings 4. Publication and citation metrics for framework documentation

6.5 Computational Complexity Analysis

6.5.1 Research Challenges

Formal analysis of CEREBRUM's computational complexity involves:

1. **Model Sizing**: Establishing complexity measures for case-bearing models of different sizes and structures.
2. **Transformation Complexity**: Analyzing the computational cost of different case transformations.
3. **Ecosystem Scaling**: Understanding how complexity scales with increasing numbers of interacting models.
4. **Optimization Techniques**: Identifying algorithmic improvements to reduce computational requirements.

6.5.2 Analytical Framework

1. **Complexity Metrics**:
   - Time complexity of case transformations as a function of model parameters
   - Space complexity of model representations in different cases
   - Communication complexity between models in different cases
   - Optimization complexity for free energy minimization processes
2. **Scaling Analysis**:
   - Analysis of complexity growth in hierarchical model structures
   - Identification of bottlenecks in large-scale model ecosystems
   - Development of approximation techniques for complex case transformations
3. **Empirical Benchmarking**:
   - Standard test cases for comparing implementation efficiency
   - Performance profiles across different model sizes and transformation types
   - Real-world scaling tests in intelligence production workflows

6.5.3 Expected Outcomes

This research direction will produce: 1. Formal complexity bounds for CEREBRUM operations 2. Optimization guidelines for implementation efficiency 3. Scaling strategies for large model ecosystems 4. Benchmarking tools for implementation comparison

6.6 Multiple Dispatch and Polymorphic Programming

6.6.1 Technical Challenges

Implementing effective multiple dispatch systems for CEREBRUM involves:

1. **Type System Integration**: Designing type systems that accurately represent case relationships and transformations.
2. **Performance Optimization**: Ensuring efficient dispatch mechanisms without runtime overhead.
3. **Language Compatibility**: Adapting to different programming language capabilities and constraints.
4. **Static Analysis**: Enabling compile-time verification of case transformation properties.

6.6.2 Implementation Approaches

1. **Pattern Matching Systems**:

```
# Pseudocode for multiple dispatch based on case
function process(model@NOM, data) -> {
  # Nominative-specific processing
}

function process(model@ACC, update) -> {
  # Accusative-specific processing
}

function process(model@DAT, input_stream) -> {
  # Dative-specific processing
}
```

2. **Trait/Interface-Based Systems**:

```
# Pseudocode for interface-based dispatch
interface NominativeCapable {
  generatePredictions(): Predictions
}

interface AccusativeCapable {
  receiveUpdates(updates: Updates): void
}

interface DativeCapable {
  processInputStream(stream: InputStream): void
}
```

3. **Dynamic Registration Systems**:

```
# Pseudocode for dynamic dispatch registry
CaseRegistry.register(NOM, ModelType, (model, context) => {
  // Nominative processing logic
});

CaseRegistry.register(ACC, ModelType, (model, context) => {
  // Accusative processing logic
});
```

6.6.3 Evaluation Criteria

Multiple dispatch implementations will be evaluated based on: 1. Expressiveness and alignment with theoretical framework 2. Performance characteristics and optimization potential 3. Type safety and compile-time guarantees 4. Developer ergonomics and learning curve

6.7 Database Methods for Case-Bearing Models

6.7.1 Technical Challenges

Developing specialized database structures for CEREBRUM involves:

1. **Schema Design**: Creating flexible schemas that can represent models in different cases.
2. **Query Optimization**: Designing efficient query patterns for case-specific operations.
3. **Transformation Storage**: Representing and storing case transformations efficiently.
4. **Consistency Guarantees**: Ensuring data consistency across case transformations.

6.7.2 Proposed Database Architectures

1. **Graph Database Approach**:
    - Models represented as nodes with case-specific properties
    - Transformations represented as typed edges between models
    - Query patterns optimized for graph traversal operations
    - Support for temporal versioning of models across transformations
2. **Document Database Approach**:
    - Case-specific model representations as nested documents
    - Transformation records as separate documents with references
    - Indexing strategies optimized for case-specific queries
    - Versioning through immutable document chains
3. **Hybrid Relational-NoSQL Approach**:
    - Core model data in relational tables with strong consistency
    - Case-specific extensions in flexible NoSQL structures
    - Transformation logs in append-only tables
    - Materialized views for frequently accessed case representations

6.7.3 Query Language Extensions

Development of case-specific query language extensions:

```
-- Example SQL-like query language with case extensions
SELECT * FROM Models
WHERE Case = NOM
  AND SUPPORTS_TRANSFORMATION_TO(ACC)
  AND FREE_ENERGY < threshold;

-- Transformation query
TRANSFORM (
  SELECT * FROM Models WHERE id = 123
) TO CASE DAT
WITH PRECISION_WEIGHTING = 0.8;
```

6.7.4 Evaluation Metrics

Database solutions will be evaluated based on: 1. Query performance for common case-related operations 2. Storage efficiency for models with multiple case representations 3.

Consistency guarantees during concurrent transformations 4. Scalability with increasing numbers of models and transformations

6.8 Cognitive Security Frameworks

6.8.1 Research Challenges

Exploring security implications of case-based systems involves:

1. **Access Control Modeling**: Designing security models based on case relationships.
2. **Transformation Security**: Ensuring secure case transformations with appropriate authorization.
3. **Information Flow Control**: Managing information flow between models in different cases.
4. **Verification Techniques**: Developing formal verification methods for security properties.

6.8.2 Proposed Security Frameworks

1. **Case-Based Access Control (CBAC)**:
   - Security permissions defined in terms of allowed case transformations
   - Role-based access mapped to case-specific operations
   - Least privilege principles implemented through case constraints
   - Auditing mechanisms for case transformation activities
2. **Information Flow Control**:
   - Formal labeling of information based on case properties
   - Flow control policies preventing inappropriate case transformations
   - Declassification mechanisms for controlled case transitions
   - Verification techniques for information flow properties
3. **Formal Verification**:
   - Model checking techniques for case transformation security
   - Static analysis tools for detecting insecure transformations
   - Runtime monitoring of case-based security properties
   - Proof assistants for verifying security theorems about case systems

6.8.3 Expected Outcomes

Research in cognitive security will produce: 1. Formal security models for case-bearing systems 2. Implementation guidelines for secure CEREBRUM deployments 3. Verification tools for case-based security properties 4. Risk assessment frameworks for model ecosystems

6.9 Interdisciplinary Research Opportunities

6.9.1 Cognitive Science Collaborations

1. **Empirical Testing**:
   - Validating case transformations against human cognitive processes
   - Investigating neural correlates of case-based reasoning
   - Developing cognitive models that incorporate case transformations
2. **Linguistic Extensions**:
   - Exploring cross-linguistic variations in case systems
   - Investigating semantic universals in case relationships
   - Developing computational models of language acquisition based on case frameworks

6.9.2 Artificial Intelligence Integration

1. **Large Language Model Integration**:
   - Extending transformer architectures with explicit case representations
   - Developing prompting techniques based on case frameworks
   - Improving reasoning capabilities through case-structured generation
2. **Multi-Agent Systems**:
   - Designing agent communication protocols based on case relationships
   - Implementing negotiation protocols using case transformations
   - Developing coordination mechanisms for agents with different case roles

6.9.3 Practical Applications

1. **Education and Training**:
   - Developing educational tools based on case transformations
   - Creating training curricula for case-based reasoning
   - Designing assessment methods for case understanding
2. **Healthcare Applications**:
   - Implementing clinical decision support systems with case frameworks
   - Developing patient monitoring systems with appropriate case assignments
   - Creating healthcare coordination systems based on case transformations

6.9.4 Research Methodology

Interdisciplinary collaboration will be structured through: 1. Joint research projects with defined deliverables 2. Shared datasets and benchmarks for evaluation 3. Regular interdisciplinary workshops and conferences 4. Collaborative publications in cross-disciplinary journals

6.10 Conclusion: A Comprehensive Research Agenda

The future directions outlined in this supplement establish a comprehensive research agenda for the CEREBRUM framework. This agenda spans theoretical developments, practical implementations, and interdisciplinary applications, providing a roadmap for researchers and practitioners to extend and apply the framework.

The successful pursuit of these directions will transform CEREBRUM from an initial theoretical framework into a comprehensive ecosystem of tools, methods, and applications that advance our understanding of cognitive modeling and intelligence production. By addressing the technical challenges and research opportunities identified here, the community can realize the full potential of case-based cognitive modeling across multiple domains and applications.

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.