Technical Literature Review and Theoretical Background for the Project *Pattern Discovery in Symbolic Visual Structures*

Table of Contents

1. Introduction to Conceptual Blending: From Cognitive Theory to Computational Creativity

The capacity to invent new concepts by combining existing ones is a hallmark of human intelligence. While classical theories of cognition often focused on static categories or compositional logic, the turn of the 21st century saw the emergence of frameworks designed to explain the dynamic, constructive nature of meaning. Foremost among these is Conceptual Blending Theory (also known as Conceptual Integration), which posits that the human mind navigates complexity not merely by retrieving stored schemas, but by actively constructing "blends", temporary mental spaces that fuse disparate elements into novel, emergent structures. This chapter traces the historical development of the theory, defines its structural taxonomy, and reviews its translation into computational systems.

1.1 Historical Origins and Theoretical Foundations

The intellectual lineage of Conceptual Blending can be traced back to Arthur Koestler's *The Act of Creation* (1964), which introduced the notion of "Bisociation", the creative leap that occurs when two previously unrelated matrices of thought are interconnected. Koestler argued that humor, science, and art all share this underlying mechanism of perceiving a situation through two self-consistent but mutually incompatible frames of reference.

In the 1990s, cognitive linguist Gilles Fauconnier and literary scholar Mark Turner formalized these intuitions into a rigorous cognitive model. Building on Fauconnier's earlier work on Mental Spaces (1985), which shifted the unit of semantic analysis from static truth-conditions to dynamic, temporary packets of working memory, they proposed that "blending" was a fundamental cognitive operation.

The theory was first articulated in technical reports such as *Conceptual Projection and Middle Spaces* (1994) and formalized in the seminal paper *Conceptual Integration Networks* (1998). It reached its fullest expression in their landmark book, *The Way We Think* (2002), which argued that blending is not reserved for high-level creativity but is the engine behind everyday cognition, from understanding "blue cup" (blending perception with object frames) to grasping complex metaphors like "The Grim Reaper" (blending death with human agency).

1.2 The Structural Mechanics of Blending

Unlike simple analogical mapping, which transfers structure from a Source to a Target, Conceptual Blending involves a multi-space network. Fauconnier and Turner defined a Conceptual Integration Network as consisting of at least four connected mental spaces:

1. Input Spaces: Two (or more) distinct mental spaces containing content from different domains (e.g., a "Boxing" space and a "Business" space).
2. Generic Space: A space containing the abstract structure shared by the inputs (e.g., "Competition" or "Agonistic interaction"). This space maps the counterparts between inputs.
3. Blended Space: The space where material from the inputs is selectively projected and integrated. Crucially, this space develops emergent structure: relations and logic that exist in neither input.

Taxonomy of Networks

The theory categorizes networks based on their topological complexity:

- Simplex Networks: One input provides a frame (roles), the other provides values. The blend is simply an instantiation (e.g., "Paul is the father of Sally").
- Mirror Networks: All spaces share the same organizing frame. These are often used for comparison or superimposition (e.g., "The Buddhist Monk Riddle," where two temporal journeys are superimposed to find a crossing point).
- Single-Scope Networks: The blend relies on the organizing frame of *one* input to structure the content of the other (e.g., "Corporate Boxing," where business events are understood strictly through the rules of boxing).
- Double-Scope Networks: The most creative form, where the blend resolves clashes between inputs by integrating structure from *both*. This creates a new category of reality (e.g., a "Computer Virus," which has biological replication but mechanical implementation).

1.3 Mathematical Formalization: Algebraic Semiotics

While Fauconnier and Turner provided the cognitive architecture, computer scientist Joseph Goguen provided the mathematical rigor necessary for implementation. In his framework of Algebraic Semiotics (1998, 2005), Goguen sought to formalize meaning construction in user interfaces and diverse symbol systems using Category Theory.

Goguen posited that "Mental Spaces" could be formalized as Algebraic Theories (signatures and axioms), and the mappings between them as Semiotic Morphisms. This allowed the imprecise notion of "blending" to be rigorously defined as a Categorical Pushout (or Colimit). In this view:

- The Generic Space is the base object in a diagram.
- The Input Spaces are objects mapped from the base.
- The Blend is the structural "sum" (colimit) that unifies the inputs while respecting the shared structure of the generic space.

Goguen's work introduced the concept of "3/2 Pushouts" to handle the partiality and messiness of human blends, where not all axioms are preserved perfectly. This formalization was instrumental in transitioning Blending Theory from a descriptive linguistic tool to a prescriptive engineering specification, directly influencing later computational projects like COINVENT.

1.4 Computational Implementations

The formal precision of Blending Theory, with its graphs, mappings, and projection rules, made it an attractive candidate for implementation in Artificial Intelligence, particularly within the subfield of Computational Creativity.

Divago (c. 2007) One of the earliest comprehensive systems was *Divago*, developed by Francisco Câmara Pereira. Divago modeled blending as a search process within a symbolic knowledge base. It utilized a "Factory" module to map concepts and a set of "Optimality Constraints" (derived from Fauconnier & Turner) to evaluate the quality of generated blends. The system successfully invented novel concepts in domains like noun-noun combinations (e.g., inventing a "Houseboat" by blending properties of shelter and mobility).

The COINVENT Project and Eppe et al. (c. 2018) A significant leap in formalization came with the European *COINVENT* project, architected by Eppe et al. (2018). This framework unified Goguen's categorical approach with Answer Set Programming (ASP) to make the blending process computationally feasible. By treating concept invention as a search problem over algebraic specifications, Eppe et al. demonstrated that a machine could autonomously compute the "Generic Space" via generalization and generate valid "Colimits" (blends) that satisfied rigorous consistency checks. This body of work verified that high-level blending could be grounded in logic programming, applying it successfully to complex domains like mathematical proof generation and musical harmonization.

Heuristic-Driven Theory Projection (HDTP) Parallel to these efforts, HDTP emerged as a framework for analogy making that relies on Anti-Unification to compute the "least general generalization" between two logical theories. While primarily focused on analogy, its mechanism for identifying the shared structure (Generic Space) overlaps significantly with the computational requirements of blending.

These implementations demonstrate that Conceptual Blending is not merely a descriptive metaphor for human thought but a computable algorithm capable of generating novel, interpretable abstractions in symbolic systems.

1.5 Generative AI and Latent Space Blending

Recent advances in Generative AI, particularly Text-to-Image (T2I) diffusion models, have introduced a new paradigm of Latent Conceptual Blending. These models demonstrate a remarkable capacity to blend concepts at the level of style, texture, and geometry. A prompt such as "an airplane that is also a fish" triggers a sophisticated blending process: the user defines the Input Spaces (Airplane, Fish) via the prompt (Composition), and the model performs the Completion and Elaboration by finding the vector-space overlap (Generic Space) between fins and wings, generating a coherent "Fish-Plane."

Methods such as "IT-Blender" (Cho et al., 2025) formalize this by modifying the self-attention layers of diffusion models (such as Stable Diffusion and FLUX) to mix the latent visual features of a reference image with the semantic features of a text prompt. This allows for automated, high-fidelity blends, such as a "Cake-Sneaker," where the visual texture of food is mapped onto the topology of a shoe.

However, a critical distinction must be made between Perceptual Blending (GenAI) and Structural Blending (Neuro-Symbolic). Neural models can be effective at Single-Scope or Mirror blends involving sensory attributes because they operate on statistical correlations in pixel/latent space. They struggle, however, with Double-Scope blends that require abstract causal logic. A diffusion model can blend the look of a grid with the look of a sorted list, but it cannot blend the physics of Gravity with the algorithm of Sorting to solve a novel reasoning task. This limitation underscores the necessity of the neuro-symbolic approach proposed in this project: to solve ARC-AGI, the agent must blend the causal mechanisms of the problem, not just its visual appearance.

1.6 Concept Blending in the Hyperon Framework (MeTTa)

The transition from classical OpenCog to OpenCog Hyperon marks a fundamental shift in how Conceptual Blending is architected within Artificial General Intelligence (AGI) systems. While earlier versions (OpenCog Classic) included blending modules within specific cognitive architectures like CogPrime or OpenPsi, these were often constrained by the rigidity of the underlying C++ and Scheme implementations, effectively creating silos between logical reasoning and creative blending.

The Neuro-Symbolic Opportunity: "Cognitive Synergy"

Hyperon is explicitly designed to support Cognitive Synergy, a design principle where distinct AI algorithms (evolutionary learning, logic, pattern mining) dynamically update a shared knowledge graph (the Atomspace). Unlike systems like *Divago* which relied on static, pre-compiled knowledge bases, Hyperon allows "Input Spaces" to be populated in real-time by perceptual agents or neural networks. This dynamic grounding is essential for tasks like ARC-AGI, where the "concepts" (e.g., a specific grid shape) do not exist in the system until the task is observed.

Hypergraphs as Mental Spaces

In Hyperon, Fauconnier's "Mental Spaces" are conceptualized as subsets or subgraphs within the Atomspace. Hyperon supports the notion of scoped or nested working contexts (conceptually similar to Mental Spaces), which future versions plan to implement via nested or modular Atomspaces. (Goertzel et al., 2023). This allows blending operations to isolate temporary atoms, enabling counterfactual reasoning ("If gravity were reversed..."), without corrupting the system's long-term memory.

Pattern Matching as Projection

The MeTTa interpreter features a unique unification engine that supports non-deterministic pattern matching. This engine serves as the computational realization of the "Cross-Space Mapping" required by Blending Theory.

- **Anti-Unification:** While not a native primitive, MeTTa's matcher can be used to implement anti-unification-like operations that identify structural correspondences between graphs. This allows the system to compute the "Generic Space" (the shared structure) on the fly. However, it is also important to note that while MeTTa provides the efficient mechanism for this unification, it does not inherently solve the search complexity. Finding the optimal 'Generic Space' remains computationally expensive. Consequently, the project's blending mechanism must rely on structural heuristics, such as restricting matches to spatially connected components, to prune the search space and prevent combinatorial explosion during the abstraction phase.
- **Gradual Typing:** MeTTa's type system is designed to support gradual and potentially dependent types (Goertzel, 2021), allowing the system to blend rigorous logical types (e.g., Integer) with fuzzy, learned types (e.g., (Similar-to-Red)). This flexibility is crucial for "Double-Scope" blends where strict type signatures often clash.

Metagraph Rewriting as Emergence

The most significant technical advancement in Hyperon is the treatment of code as data. MeTTa programs are themselves represented as atoms within the metagraph. This allows for Self-Modifying Code, where the output of a blending operation is not just a static data point (like "Houseboat") but a new executable function. In the context of ARC-AGI, this allows the system to blend a "Visual Pattern" (Input A) with a "Transformation Rule" (Input B) to rewrite its own source code, effectively generating a new solver algorithm for the specific task at hand. This aligns with the "Refactoring" step in *DreamCoder*, but implemented via direct metagraph rewriting rather than external program synthesis.

2. The Domain of Application: The ARC-AGI Benchmark

2.1 Defining Intelligence as Skill-Acquisition

The Abstraction and Reasoning Corpus (ARC-AGI), introduced by François Chollet in *On the Measure of Intelligence* (2019), represents a paradigm shift in AI evaluation. Chollet argues that "skill" (performance on a static task) is a poor proxy for intelligence, as it can be bought with unlimited priors (hard-coding) or unlimited training data (memorization). Instead, he defines intelligence as skill-acquisition efficiency, the ability to adapt to novel tasks using a minimal number of examples.

ARC operationalizes this definition by presenting tasks that are strictly novel (test tasks share no specific logic with training tasks) and are designed to rely on basic structural priors rather than task-specific training. These include assumptions about:

- Objectness: The persistence of distinct entities
- Goal-directedness: The interpretation of transformations as purposeful or rule-governed.
- Geometry & Topology: Symmetry, rotation, containment, connectivity.

2.2 The "Visual Intelligence" Gap

While ARC data is technically static (JSON pairs of 2D matrices), the reasoning required is often dynamic and temporal. Tasks frequently depict kinematic processes such as "gravity," "bouncing," or "growing." A major limitation of current symbolic solvers is their treatment of the grid as a static array of pixels. Following the "Buddhist Monk" strategy in cognitive science (Fauconnier & Turner), successful reasoning often requires visual compression: converting temporal sequences (implied motion) into spatial representations (SpaceTime atoms). This allows an agent to identifying a "trajectory" not as a simulation step, but as a static geometric object (e.g., a line in 3D space-time) that can be matched and blended.

2.3 Current State-of-the-Art Solvers

Attempts to solve ARC have largely bifurcated into two approaches, neither of which fully achieves the "abstraction" goal:

- **Discrete Program Search (e.g., Icecuber):** The 2020 Kaggle winner used a brute-force search over a handcrafted DSL of ~100 primitives. While effective for simple tasks (~20% accuracy), this approach scales poorly ($O(N!)$) and lacks the ability to learn new concepts; it can only find what is already expressible in its fixed DSL.
- **LLM-Guided Synthesis (e.g., Greenblatt/Redwood):** Recent SOTA approaches (~42%) utilize Large Language Models (like GPT-4) to generate thousands of Python programs. This essentially puts the "Wake" phase of program synthesis on steroids, using the LLM's vast training data as a heuristic guide. However, these systems lack a "Sleep" phase; they do not learn from their own solutions to compress the library, meaning they start from scratch on every new task.
- **Test-Time Training (TTT) (e.g., Cole & Osman, 2025):** This approach fine-tunes the weights of a neural network on the fly using the specific task examples. While closer to "learning to learn," the resulting abstractions are opaque weights rather than interpretable symbolic concepts.

3. Mechanisms of Abstraction: Program Synthesis & Library Learning

To bridge the gap between static search and opaque neural intuition, this project looks to Inductive Program Synthesis, specifically frameworks that learn their own Domain Specific Languages (DSLs).

3.1 The Combinatorial Explosion Problem

The fundamental bottleneck of program synthesis is combinatorial explosion. As the complexity of a required program increases, the search space grows exponentially. A task requiring a complex "sorting" algorithm is impossible to find by brute force if the language only contains primitive "swap" and "compare" operations. To solve complex tasks, the system must possess high-level abstractions (like sort or filter) that compress the solution length.

3.2 The DreamCoder Framework: Bootstrapping Abstractions

DreamCoder (Ellis et al., 2021) offers a robust architectural solution to this problem. Unlike traditional synthesis which relies on a static DSL, DreamCoder treats the DSL as a learned variable. It operates on a "Wake-Sleep" cycle inspired by the Helmholtz Machine:

- **Wake Phase (Program Synthesis):** The system attempts to solve tasks using its current library. Initially, it can only solve easy tasks using verbose, inefficient code (e.g., manually writing out recursion logic for a simple loop).
- **Sleep Phase (Abstraction & Library Learning):** The system analyzes the successful programs from the Wake phase to find repeated structures. Using a refactoring algorithm based on Anti-Unification (data structure alignment), it identifies common subgraphs across different programs.
    - Example: If the system solves a "doubling" task and a "decrement" task using the same recursive structure, it extracts that structure, parameterizes the difference, and invents a new function (effectively re-discovering map or fold).
- **MDL Objective:** The system is guided by the Minimum Description Length (MDL) principle, seeking to minimize the sum of the library size + the solution lengths.

This "bootstrapping" process allows the system to iteratively solve harder tasks by building a ladder of abstractions, using the solution to Task A to learn the tool needed for Task B.

3.3 Relevance to Concept Blending

DreamCoder validates the core hypothesis of this project: that high-level patterns can be discovered by compressing the symbolic representation of successful reasoning traces. The "Refactoring" step in DreamCoder is mechanistically isomorphic to identifying the "Generic Space" in Conceptual Blending. By implementing this architecture within Hyperon/MeTTa, we aim to extend this capability from pure code refactoring to Neuro-Symbolic Blending, where the "abstractions" are not just functions, but multimodal concepts uniting visual topology with causal logic.

4. Summary

This review has explored how Conceptual Blending and Inductive Program Synthesis both address the problem of building reusable abstractions for ARC-style reasoning tasks. DreamCoder shows that abstractions can be discovered by compressing successful solution programs. Conceptual Blending provides a complementary perspective, focusing on how new structural concepts can emerge by combining and generalizing existing representations. Together, these perspectives motivate the investigation of abstraction mechanisms within a symbolic and neuro-symbolic framework such as OpenCog Hyperon.

The goal of this document is not to provide a comprehensive survey of the field. Instead, it outlines the technical background needed to frame the experimental scope of the project and to justify key design decisions. This review should be seen as a working document that may evolve as the project progresses.

The author has used AI-assisted literature exploration tools to help identify relevant work and organize the material. All generated content has been reviewed and checked for accuracy to the best of the author's knowledge. Constructive feedback and discussion are welcome in order to further refine both the theoretical framing and the practical implementation of the project.

References

Cho, W., Zhang, Y., Chen, Y.-Y., & Inouye, D. I. (2025). Imagine for Me: Creative Conceptual Blending of Real Images and Text via Blended Attention. *arXiv preprint arXiv:2506.24085.*
https://arxiv.org/abs/2506.24085
https://imagineforme.github.io/

Chollet, F. (2019). On the measure of intelligence. *arXiv preprint arXiv:1911.01547*.
https://arxiv.org/abs/1911.01547

Cole, J., & Osman, M. (2025). *MindsAI ARC Solution: Test-Time Training and AIRV* [Source code]. GitHub.
https://github.com/jcole75/arc_2025_mindsai

Ellis, K., Wong, C., Nye, M., Sablé-Meyer, M., Morales, L., Hewitt, L., ... & Tenenbaum, J. B. (2021, June). Dreamcoder: Bootstrapping inductive program synthesis with wake-sleep library learning. In *Proceedings of the 42nd acm sigplan*

international conference on programming language design and implementation (pp. 835-850). https://dl.acm.org/doi/pdf/10.1145/3453483.3454080

Eppe, M., Maclean, E., Confalonieri, R., Kutz, O., Schorlemmer, M., Plaza, E., & Kühnberger, K. U. (2018). A computational framework for concept blending. *Artificial Intelligence*, 256, 105–129. https://www.sciencedirect.com/science/article/pii/S000437021730142X

Fauconnier, G. (1985). *Mental Spaces: Aspects of Meaning Construction in Natural Language*. MIT Press.

Fauconnier, G., & Turner, M. (1994). *Conceptual projection and middle spaces* (UCSD Cognitive Science Technical Report 9401). Department of Cognitive Science, University of California, San Diego.

Fauconnier, G., & Turner, M. (1998). Conceptual integration networks. *Cognitive Science*, 22(2), 133–187.

Fauconnier, G., & Turner, M. (2002). *The Way We Think: Conceptual Blending and the Mind's Hidden Complexities*. Basic Books.

Goertzel, B., et al. (2023). OpenCog Hyperon: A Framework for AGI at the Human Level and Beyond. *arXiv preprint arXiv:2310.18318*.

Goertzel, B. (2021). Reflective Metagraph Rewriting as a Foundation for an AGI "Language of Thought". *arXiv preprint arXiv:2112.08272*.

Goertzel, B., Iklé, M., et al. (OpenCog Hyperon design documents)

Goguen, J. A. (1998). An introduction to algebraic semiotics, with application to user interface design. In *Computation for Metaphors, Analogy, and Agents* (pp. 242–291). Springer.

Goguen, J. A. (2005). What is a concept? In *Conceptual Structures: Common Semantics for Sharing Knowledge* (pp. 52–77). Springer.

Greenblatt, R. (Redwood Research). (2024). *Getting 50% (SoTA) on ARC-AGI with GPT-4o*. https://blog.redwoodresearch.org/p/getting-50-sota-on-arc-agi-with-gpt

Icecuber. (2020). *ARC 1st place solution*. Kaggle. https://www.kaggle.com/c/abstraction-and-reasoning-challenge/discussion/154597

Koestler, A. (1964). *The Act of Creation*. Macmillan.

Pereira, F. C. (2007). *Creativity and Artificial Intelligence: A Conceptual Blending Approach*. Mouton de Gruyter.

Schwering, A., Krumnack, U., Kühnberger, K. U., & Gust, H. (2009). Syntactic principles of heuristic-driven theory projection. *Cognitive Systems Research*, 10(3), 251-269.