

Article

Embrace LLM-Based Cognitive Architecture to Boost Team Problem-Solving in Open-Ended Tasks

Hashmath Shaik , Gnaneswar Villuri  and Alex Doboli * 

Department of Electrical and Computer Engineering, Stony Brook University, Stony Brook, NY 11794-2350, USA; hashmath.shaik@stonybrook.edu (H.S.); gnaneswar.villuri@stonybrook.edu (G.V.)

* Correspondence: alex.doboli@stonybrook.edu

Abstract

Open-ended, team-based problem solving demands (i) a bridge between stochastic language models and symbolic control, (ii) mechanisms for idea elaboration, (iii) feature-level concept combination, and (iv) internal representations that support understanding beyond mere association. We present a cognitive architecture (CA) that couples an LLM with an editable knowledge-graph (KG) scaffold and a controller that adaptively schedules five reasoning strategies. Elaborations are cast as graph updates validated against coverage and consistency checks; combinations produce property- and relation-level recompositions. On 30 collaborative programming dialogs (nine representative scenarios), adaptive prompting improves solution completeness by 19.1% and reduces required turns by 18.5% over a CoT baseline; explicit concept combinations increase Distinct-3 by 12.4 points with a +0.7 gain in human-rated creativity. Ablations show that Soft→Pruning scaffolds best support early elaboration, while Hard partitioning helps under ambiguity. The CA demonstrates a practical route to aligning LLMs with team intent in open-ended tasks.

Keywords: open-ended problem solving; large language models; cognitive architecture; knowledge graph; team collaboration

1. Introduction

Solving open-ended problems requires devising new solutions with a significant departure and characteristics from previous implementations, hence, relying on precedence (like in traditional design) is of limited usefulness [1,2]. Such problems describe requirements that can be satisfied by a variety of implementations with diverse functionality and performance. In addition to employing an effective solving strategy, open-ended problem solving requires activities, e.g., understanding the problem requirements, and having mechanisms for aligning a solution to the problem description, establishing the expectations for a given implementation, and evaluating the degree to which a solution meets the requirements [3]. This paper focuses on devising problem-solving strategies while assuming the existence of the other mechanisms, although research on each of these related activities is also needed. While open-ended problems are ubiquitous in healthcare, education, environment protection, and so on, their solving process remains difficult with virtually no computational support [4].

Open-ended problem solving introduces new challenges in addition to the well-known difficulties of understanding problem solving, a research topic studied for a long time [5–7]:

- **Importance of collaborative work in teams:** Open-ended problems are often complex and cross-disciplinary in nature. Hence, their solving requires collaborative work



Academic Editor: Vladimír Bureš

Received: 12 January 2026

Revised: 23 February 2026

Accepted: 13 March 2026

Published: 16 March 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

in teams, where spoken dialog is a main instrument for collaboration [8]. Unlike traditional Natural Language Processing (NLP) research, which considers curated text, spoken dialog is often grammatically incorrect, ambiguous, imprecise, and logically only partially correct [9].

- **Pursuing the intentions, preferences, and solving flows described by the human teams:** Current ML software, especially LLMs, can often generate solutions for a variety of well-defined problems without any human input, like solving math problems or creating code for a specification. In other cases, the LLM output significantly deviates from the intentions of the inputs provided by humans. However, in the absence of precise criteria of what an effective solution means, subjective intentions and preferences are critical, and thus the ML support should assist a team rather than replace it. Other kinds of interactions are needed between humans and LLMs.
- **Extending generative capabilities beyond similarities and analogies:** It can be argued that current ML approaches learn some form of a (complex) regression model using the training data without actually understanding how the data features and relations (among data) must be linked together in problem solving. This arguably limits solution generation to cases that are similar or analogous to problems that were previously solved, thus captured by the training samples. Solutions to open-ended problems bring important changes as compared to already existing solutions.

This paper argues that current ML approaches and algorithms are insufficient to address the three types of challenges. Instead, research must focus on the following four aspects:

1. **Bridging the stochastic vs. symbolic modeling gap:** Current LLMs use stochastic modeling and learning [10], but evidence in cognitive psychology suggests that problem solving employs symbolic representations [4]. While embeddings and modeling elements (attention, normalization layers, and so on) are numerical, and LLM outputs are symbolic (i.e., word sequence, math formulas, code, etc.), the meaning of a symbol (e.g., word) is closely tied to the symbol's sentence and context without having the autonomy of a concept as defined in cognitive psychology, i.e., defining features, characteristic exemplars or specific categories [11,12]. This paper argues that autonomy can be retrieved at least partially by using structural representations of sentences to reflect their hierarchical nesting and word roles, as well as adaptive prompting to solve inconsistencies between the representations.
2. **Idea elaboration:** Elaborating concepts, relations, and features into progressively deeper solutions is central to open-ended problem solving, yet current generative ML approaches (GANs, transformers, and LLMs) offer limited support for this activity when used in isolation. Because these methods rely on regression-based modeling, where training samples determine the weights of feedforward perceptrons, attention, and normalization layers [10], they are well suited to finding similar solution points or producing summaries but tend to plateau rather than “drill” towards more in-depth solutions as required by elaboration. Traditional “generative” domains, like Electronic Design Automation (EDA) for the synthesis of electronic circuits and systems, utilize a library of pre-characterized basic building blocks (BBs), like fundamental circuits, to assemble them guided by specifications describing parameterized structures made of the BBs. The parameters of the structures optimize the performance of the solutions, like speed and power consumption [13–15]. Nevertheless, the approach of creating a new solution based on assembling statically characterized BBs based on structural descriptions is unrealistic for dialog-based problem solving, as human dialog does not incorporate the formal requirements of a specification language and cannot address situations in which a structural description might not be available. Besides, trial-and-

error, a well-known method for solving open-ended problems, cannot be conducted in this approach. This paper suggests that the elaboration limitation can be addressed by pairing an LLM with external scaffolding, specifically an editable knowledge graph and an adaptive controller, that decomposes elaboration into a managed sequence of prompts and validation steps. Problem-solving strategies should mix top-down and bottom-up steps guided by adaptively selected goals. Five strategies are introduced to cover a broad range of solving scenarios.

3. **Combination of concepts:** Concept combinations have been intensely researched in cognitive psychology [1,16]. They are of three types: (i) property-based combinations transfer features from one concept to another concept, (ii) relation-based combinations introduce a new relation between two concepts, and (iii) hybrid combinations show the characteristics of both property-based and relation-based combinations [17]. For example, quantum optimization describes a property-based combination as it includes the features of quantum probabilities and entanglement into an optimization method, and the structural schematics of an embedded system for ambient monitoring utilizes relation-based combinations to link the BBs of the schematics depending on how the output of a block serves as an input for another block [2]. Hybrid combinations define a quantum power grid, i.e., using quantum principles in controlling the operation of the BBs of a power grid. Currently, there are arguably no computational methods to implement concept combinations in the sense they are defined in cognitive psychology, despite some attempts to assimilate relation-based combinations as functions that use concepts as their parameters [3]. However, this interpretation cannot replicate the generative effect of concept combinations, in which the meaning of a combination cannot be distinguished from the identities of the involved concepts, but instead is defined by them. Moreover, the concepts are perceived as atomic (indivisible) entities in functions, while concept combinations can break the concepts apart to consider or transform features.
4. **Understanding vs. associative learning:** Current ML methods perform different kinds of associative learning in which predictions are made about expected outcomes but without understanding the reasons that produced the outcomes. In addition to acquiring knowledge, understanding also involves placing the learned knowledge into an existing knowledge structure and predicting its role in different situations. Creating an internal representation during understanding is essential. This paper suggests building an internal representation similar to that used for elaboration. Early preliminary work on this idea is presented in [18].

A novel Cognitive Architecture (CA) based on an LLM is presented to implement the four ML needs. To evaluate the CA empirically, the paper is guided by three research questions:

1. **RQ1:** Does adaptive prompting significantly improve solution quality compared to a standard chain-of-thought baseline?
2. **RQ2:** How effectively does the CA utilize different cognitive strategies based on task complexity?
3. **RQ3:** What quantitative improvements are achieved through explicit concept combination?

The paper has the following structure. Section 2 summarizes related work, followed in Section 3 by a new theoretical model to address the four aspects. Section 4 discusses the CA, and Section 5 presents the empirical validation.

2. Related Work

Solving open-ended problems in real time especially through spoken, cross-disciplinary dialogue requires machinery that (i) fuses stochastic inference with symbolic manipulation, (ii) supports incremental idea elaboration, (iii) generates genuinely novel concept combinations, and (iv) grounds learning in causal world models. Four strands of research map directly onto these needs; we review each in turn, highlighting representative advances and the gaps that still remain.

1. Bridging stochastic and symbolic representations: neuro-symbolic hybrids.

A new wave of *differentiable* theorem provers and logic-aware transformers embeds explicit symbolic constraints inside large language models (LLMs). Representative systems include DeepProbLog [19], LogicT5 trained on the LogicBench corpus [20], and relational NeSy Markov Models [21]. End-to-end deductive architectures such as LOGIPT [22] learn to perform formal inference internally, obviating external provers. On the probabilistic side, tractable symbolic backbones such as probabilistic circuits [23] and NeSy Collaborative Distillation [24] push chain-of-thought accuracy above 90% on GSM8K while remaining GPU-friendly. Yet these pipelines are almost always *offline*: they pre-train a symbolic scaffold once and freeze it, limiting their capacity to grow new symbols on-the-fly or reconcile conflicting parses in noisy dialogue. Program synthesis and probabilistic programming tools, such as MultiPProver [25], interleave neural search with symbolic program induction, but they still target well-formed specification languages rather than free speech. Recent neuro-symbolic stacks, including AllegroGraph v8's KG-LLM runtime [26] and Meta's Large Concept Models [27], illustrate how mutable graph memories can steer generative models during inference, yet they stop short of delivering the dialog-sensitive reconciliation that our adaptive scaffold provides. By comparison, AlphaGeometry attains Olympiad-level geometry performance by exhaustively synthesizing roughly 10^8 theorem–proof pairs through symbolic self-play [28], highlighting how brute-force accumulation diverges from our emphasis on editable, interpretable scaffolding.

2. Idea elaboration via interactive prompting and tool use.

Prompt-engineering recipes such as Chain-of-Thought [29], ReAct [30], Plan-and-Solve [31], Tree-of-Thought search [32] show that *externalising* intermediate reasoning boosts factuality and robustness. Follow-up methods such as Self-Consistency [33] and Auto-CoT [34] sample diverse chains and take a majority vote. These strategies typically employ fixed prompt templates; adapting to mid-session goal changes by a human team is possible through further prompt engineering, but has not yet been demonstrated systematically in open-ended collaborative settings. Multi-agent tool frameworks such as Voyager [35], ChatDev [36], SWE-agent [37], and AutoGen [38] orchestrate swarms of specialised LLM agents that call APIs, execute code, and write unit tests. Their flexible memory and prompting mechanisms could, in principle, be extended to support open-ended collaboration; however, current deployments have primarily been evaluated on well-defined software-synthesis benchmarks (e.g., unit-test coverage) rather than on tasks where human intentions evolve during the dialog. SagaLLM [39] takes a step further with layered context, planning, and validation agents that reduce cascading errors relative to monolithic prompt-chains, though it still relies on fixed memory abstractions without the working/episodic differentiation explored in the present work.

3. Concept combination and compositional generalisation.

Cognitive psychology distinguishes *property-based*, *relation-based*, and *hybrid* noun–noun combinations [17]. Early machine-learning approaches used linear composition in static embedding spaces such as word2vec and GloVe [40,41], enabling analogy-style vector

arithmetic yet unable to move individual features from one concept to another. Tensor-factorisation models, for instance RESCAL [42], introduced relational structure, while recursive matrix–vector networks [43] treated predicates as higher-order tensors. Most recently, diffusion-based “concept algebra” methods explicitly decompose latent representations into editable slots and then remix them; slot-wise concept editing now achieves near-perfect zero-shot accuracy on CLEVR visual-reasoning benchmarks [44–46]. Despite this progress, current systems still depend on synthetic supervision and have seen limited exploration in open-ended dialogue.

4. From associative prediction to internal understanding.

Retrieval-augmented generation (RAG) pipelines such as Self-RAG [47], HippoRAG [48], and SeRTS [49] couple adaptive retrieval with self-reflection so that models ground claims in verifiable passages, reducing hallucinations by up to 60%. Yet RAG corrects *after the fact* rather than employing an internal causal world model. Alignment methods like reinforcement learning from human feedback (RLHF) [50], reinforcement learning from AI feedback (RLAIF) [51], and direct preference optimisation (DPO) [52], fine-tune LLMs toward human preferences without sacrificing fluency, but the rewards remain behavioural and do not compel the model to expose the mechanisms that produce its answers. First-generation agents that combine latent world models with language planning, such as Web Agents with World Models [53] and Language-Guided World Models [54], suggest that structured causal abstractions could unlock deeper understanding, though results are still preliminary and restricted to simplified environments.

Despite deep progress in each area, gaps remain. Table 1 summarizes how well each strand addresses the four core needs. No single strand closes all four gaps. Taken together, however, the literature points toward a *layered* cognitive architecture in which (i) a neuro-symbolic backbone provides persistent variables, (ii) adaptive prompting agents elaborate ideas in dialogue, (iii) feature-level combiners enable concept recomposition, and (iv) a causal world model grounds preferences learned through reinforcement-style feedback. Sections 3 and 4 translate these insights into a concrete design and research agenda.

Table 1. Coverage of prior research lines against the four core needs, based on the authors’ qualitative assessment of the cited literature. Symbols: ✓ = the research line’s stated contributions fully address the need; *partly* = partially addresses; ✗ = open gap (not addressed).

Research Line	Need 1 Stochastic– Symbolic Gap	Need 2 Idea Elaboration	Need 3 Concept Combination	Need 4 Understanding vs. Association
Prompt engineering (CoT, ReAct, etc.)	✗	<i>partly</i>	✗	✗
Retrieval-augmented generation (RAG)	✗	✗	✗	<i>partly</i>
Preference tuning (RLHF, RLAIF, DPO)	✗	✗	✗	<i>partly</i>
Neuro-symbolic hybrids (DeepProbLog, LogicT5, etc.)	<i>partly</i>	✗	<i>partly</i>	<i>partly</i>
Multi-agent tool use (Voyager, ChatDev, SWE-agent)	✗	<i>partly</i>	✗	✗
Causal representation learning	✗	✗	✗	<i>partly</i>
Proposed CA	✓	✓	✓	✓

3. Theoretical Model

The goal of problem solving is to bridge the semantic gap between problem description and solution (implementation) by conducting a sequence of steps that identify, create, combine, and assess concepts that form a solution. The presented work considers five strategies that were empirically identified by analyzing a large corpus of designs [3]. The five strategies were considered sufficient to build any problem-solving method by mixing the strategies to create effective solutions for an open-ended problem, like solutions that are high performance, novel, robust, cost-effective, and so on.

Cognitive psychology describes a concept as a set of shared features that cluster similar objects, ideas, and experiences together, typical prototypes, specific examples, or defining features [11,12]. The concepts to be included into a solution can be found in two ways: (i) recalling from memory already existing concepts that are good candidates to be used in the implementation and (ii) inventing new concepts through a trial-and-error activity that constructs new entities with distinguishing features, e.g., physical properties and possible functionality within the goal of an open-ended problem.

The concept representation of the model, called Knowledge Graph (KG) representation, is a way of organizing concepts based on their expected usefulness in solving a broad range of open-ended problems, including current and future problems. The representation includes several simultaneous, yet orthogonal ways of concept organization to serve different purposes:

- **Description of the problem space vs. description of the solution population space:** The KG representation is used to describe both the problem space and the solution population space, as shown in Figure 1b, as relating the two spaces serves to guide the selection of the five strategies.
- **Similar structure vs. role in achieving a goal:** The concepts are organized on the basis of the similarity of the structural organization of their features as a way to predict the similarity of their behavior. Concepts are also grouped based on their roles towards realizing a goal, e.g., their co-occurrence (associativity), causality, and support (enabling) of specific goals.
- **Generic vs. problem-specific memory:** The long-term memory stores the concepts in a problem-agnostic way, while the working memory has two components, the epistemic memory that includes concepts to be used in a problem-specific situation and the short-term memory to store the concepts utilized in the local, narrow confines of a single step.

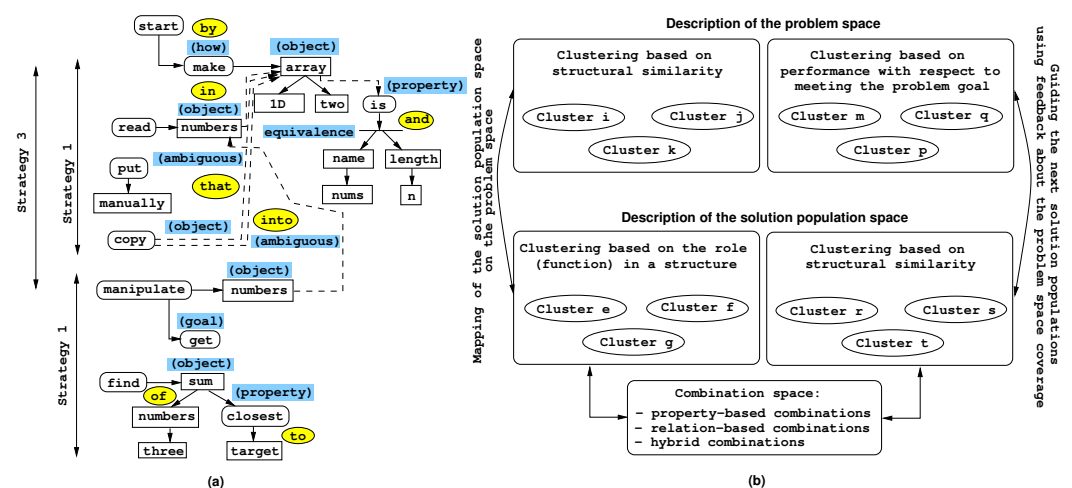


Figure 1. (a) Knowledge Graph (KG) representation used by the theoretical model and Cognitive Architecture and (b) Mapping the solution population space to the problem space.

Specifically, the Knowledge Graph (KG) representation is a hierarchical structural description that includes concepts, relations among concepts, and features of the concepts and relations. Figure 1a illustrates the KG representation that corresponds to the following dialog between two participants labeled as P1 and P2: P1: Start by making two 1D arrays. We have to manually put in the array. P2: You have to read the numbers in the array. P1: The array is named nums and length is n. You copy that array into another array. P2: So, you are manipulating the numbers to get as close as you can to target? The

KG representation must correctly describe the meaning (semantics) of the dialog including the description of the associative and causal relations between words and sentences.

The KG representation includes nouns (e.g., array, numbers, target, etc.), verbs (i.e., start, make, put, read, etc.), and features (e.g., three, n, manually, etc.). The role of sentences, like *goal* and detailing on *how* a more general idea is realized as well as the role of the words in a sentence, such as the type of verbs, like action, property, identity, and roles of nouns, i.e., objects, are highlighted in blue color. Given their importance in deciding the role, close-world words, highlighted in yellow color in the figure, are recognized using a Multi-layered Perceptron network (MLP) [9]. The MLP also recognizes the hierarchical structure of the sentences as well as words that have the same or alternative roles. The first category is named *equivalence* classes and introduced by words, like *and*, and the second class is called *alternative* and is defined by words, such as *or*. Finally, dashed lines present ambiguities and unknowns in the sentences, such as *whether*, for example, the source of copying is from the array named *nums* or if the array is actually the target.

This work assumes that any problem-solving method is a mixture of the following five strategies:

- **Strategy 1** describes an elaboration process in which a conceptual description is detailed by indicating its realization (implementation) in terms of more precise nouns and their relations. As it becomes more precise, an implementation is considered to be at a lower abstraction level than its related conceptual description. For example, Figure 1 presents the elaboration of the sentence *Start* by making two 1D arrays by reading, manually putting values, and copying an array.
- **Strategy 2** creates a new idea by combining ideas that pertain to elaborations of different conceptual ideas, each of the elaborations being produced by using Strategy 1. For example, in Figure 1a, a new elaboration is obtained by combining the elaborations *We have to manually put in the array* and *find the sum of three numbers closest to target*. Each of the two combined elaborations pertains to a different conceptual sentence.
- **Strategy 3** indicates the creation of a new conceptual description through the combination of two existing conceptual combinations. For example, in Figure 1a, Strategy 3 is used to combine two conceptual ideas, *Start by making an array* and *manipulate the numbers*. The two ideas are connected through relation-based combinations as the result of the first idea is used by the second idea.
- **Strategy 4** presents solving steps defined by exclusions rather than a specified sequence.
- **Strategy 5** describes a bottom-up solving process during which a set of steps describing detailed implementations is used to form a more abstract conceptual idea for which the initial steps serve as an elaboration.

The solving of an open-ended problem corresponds to sequences of ideas pertaining to the five strategies, which are switched under the control of the modules to align the solution to the problem description and then evaluate the degree to which a solution meets the requirements.

The completeness of the five strategies can be argued based on the fact that traditional problem-solving strategies can be expressed using the five strategies, like analogy, top-down and bottom-up design, divide and conquer, trial and error, and differential design.

4. LLM-Based Cognitive Architecture

This paper proposes a Cognitive Architecture (CA) as a mechanism to generate the sequence of LLM prompts, so that the obtained LLM responses serve to create solutions to an open-ended problem. In this approach, the LLM is a comprehensive repository of

knowledge that is organized in unknown ways and which was trained without a specific domain or problem in mind, hence serving a general purpose. Getting problem-specific LLM responses that are then used to assemble problem solutions and converge towards an effective implementation requires that the sequence of prompts adaptively use the sensitivity of satisfying the problem needs with respect to the selected solution structures, their concepts, and features in exploring the solution space. For example, if a specific problem requirement is not met, then concepts and features that are more related to the requirement are given higher preference during exploration. To offer capabilities beyond greedy exploration, the CA offers capabilities for trial-and-error and idea combination along the five strategies presented in Section 3. The LLM is the long-term, general-purpose memory of the CA, while a KG representation realizes the problem-dependent memory including the short-term memory and the epistemic memory.

The KG scaffold is not a static schema but an editable control plane that gates how prompts are issued, validated, and stored. Inspired by recent KG-LLM co-processing stacks [26,27], we partition the scaffold into (i) a persistent concept graph that captures reusable abstractions and (ii) task-scoped working memories that record episodic dialog states. This partition enables context-sensitive elaboration without forcing the LLM to relearn stable primitives while also giving downstream verifiers interpretable hooks for overrides.

Algorithm 1 summarizes the controller loop. Each dialog turn operates on a tuple $(\mathcal{G}_{\text{perm}}, \mathcal{G}_{\text{task}}, \Pi)$, where $\mathcal{G}_{\text{perm}}$ is the persistent concept graph, $\mathcal{G}_{\text{task}}$ is the task-scoped working memory, and Π is a queue of active goals. The controller alternates between goal expansion and validation:

- (i) **Goal selection.** A scoring function $s(g)$ combines concept-coverage heuristics with user-stated priorities to pick the next goal $g \in \Pi$.
- (ii) **Prompt synthesis.** The planner builds a prompt template by selecting, from $\mathcal{G}_{\text{task}}$, the neighborhood of nodes that support g , concatenating their summaries, and appending a strategy directive (e.g., elaborate, combine).
- (iii) **LLM inference.** The LLM produces a candidate response r together with self-reported confidence tokens.
- (iv) **Grounding & updates.** A deterministic parser projects r into KG triples. Edges whose semantic similarity exceeds τ_{add} are added to $\mathcal{G}_{\text{task}}$; contradictory edges trigger a revision routine that either re-prompts with a clarification request or raises a conflict for human review.
- (v) **Validation.** Heuristics check that new triples improve coverage of the problem specification and that the resulting subgraph remains connected. Failed checks push follow-up goals into Π .

The key subroutines in Algorithm 1 operate as follows. GOALIFY decomposes the problem brief $\mathcal{B}$ into an initial set of sub-goals by extracting noun phrases and requirement predicates from the parsed input. SYNTHESIZEPROMPT retrieves the local neighborhood of the target goal g in $\mathcal{G}_{\text{task}}$, serialises the relevant nodes and edges as natural-language context, and appends the strategy directive (e.g., elaborate or combine) selected by the scoring function s . PARSETOTRIPLES applies the spaCy dependency parser to the LLM response r and maps the resulting predicate–argument structures to candidate KG triples. UPDATEGRAPH computes the pairwise semantic similarity between candidate triples and existing edges; triples above the addition threshold τ_{add} are inserted, while edges that contradict existing triples beyond the revision threshold τ_{rev} are queued for re-prompting. VALIDATE checks that the updated subgraph improves coverage of the problem specification and remains connected; failed checks generate repair goals via REPAIRGOALS.

Algorithm 1 Controller for adaptive prompting and KG maintenance

```

1: Input: problem brief  $\mathcal{B}$ , strategies  $S_{1...5}$ 
2: initialize  $\mathcal{G}_{perm}$  from domain memory,  $\mathcal{G}_{task} \leftarrow \text{SEEDGRAPH}(\mathcal{B})$ ,  $\Pi \leftarrow \text{GOALIFY}(\mathcal{B})$ 
3: while  $\Pi \neq \emptyset$  do
4:    $g \leftarrow \arg \max_{g' \in \Pi} s(g')$ 
5:    $\text{prompt} \leftarrow \text{SYNTHESIZEPROMPT}(g, \mathcal{G}_{task}, S_{1...5})$ 
6:    $r \leftarrow \text{LLM}(\text{prompt})$ 
7:    $\mathcal{T} \leftarrow \text{PARSETOTRIPLES}(r)$ 
8:    $\mathcal{G}_{task}, \Delta \leftarrow \text{UPDATEGRAPH}(\mathcal{G}_{task}, \mathcal{T}, \tau_{add}, \tau_{rev})$ 
9:    $\Pi \leftarrow \Pi \cup \text{SPAWNGOALS}(\Delta, S_{1...5})$ 
10:  if  $\text{VALIDATE}(\mathcal{G}_{task}, g)$  fails then
11:     $\Pi \leftarrow \Pi \cup \text{REPAIRGOALS}(g)$ 
12:  end if
13:  remove  $g$  from  $\Pi$ 
14: end while
15: return  $\mathcal{G}_{task}$ 

```

Figure 2a details the proposed mechanism to bridge the semantic gap between the open-ended problem description and the generation of a pool of solutions that solve the problem. The mechanism relies on adaptively prompting the LLM to produce responses, either more elaborated or more conceptual, which are then related to each other to create either property-based (e.g., concepts (modules) with new features) or relation-based combinations (i.e., concepts connected through functional relations). The novel parts of the CA are highlighted in yellow in the figure:

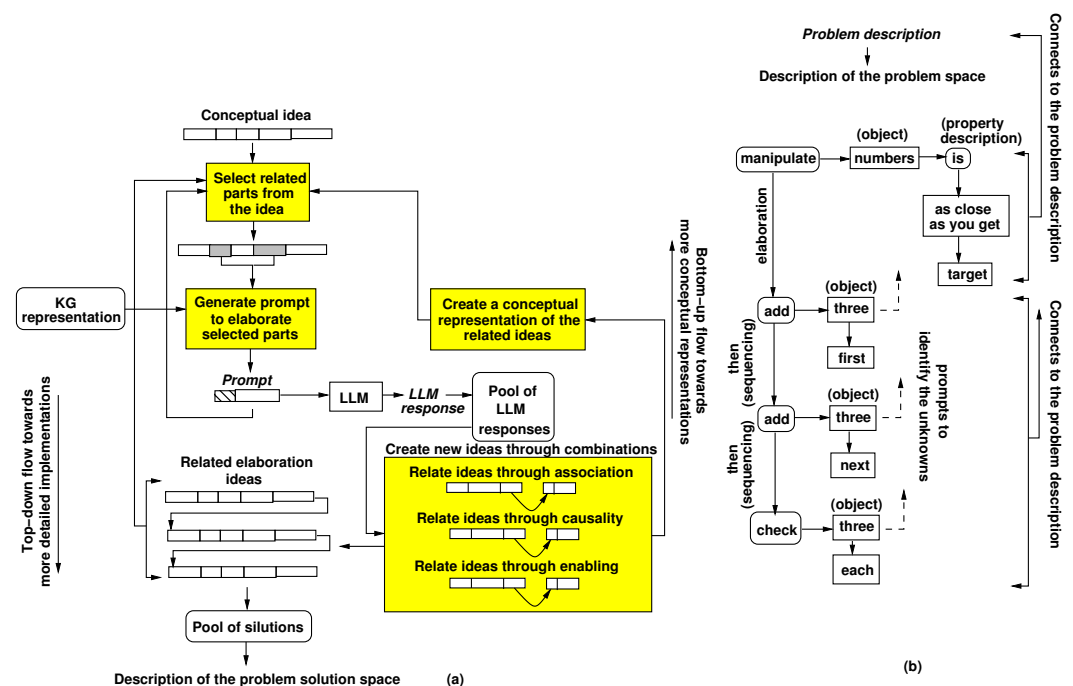


Figure 2. (a) Main principle of the LLM-based Cognitive Architecture and (b) idea elaboration during problem solving.

1. **Generate prompts to elaborate the selected parts:** This algorithm is part of the top-down flow that elaborates the selected concepts, relations, and features by adding details towards the structural representation of a solution.
2. **Create a conceptual representation of the related ideas:** This activity produces a higher-level representation of a sequence of related sentences. It is used in bottom-up

reasoning, including trial-and-error, when trials at low levels lead to insight at a more conceptual level.

3. **Create new ideas through combinations:** This step creates new ideas by connecting existing ideas or their fragments according to property-based and relation-based combinations, so that the new ideas are useful in problem solving.

The three parts are described below. **1. Generate prompts to elaborate the selected parts** These prompts are used to elaborate a more abstract conceptual idea into an idea that is more precise, hence closer to an implementation (solution). Hence, an elaboration step details the realization of a concept, relation, or features (properties) of a concept. Different elaboration alternatives can be created for the same higher-level concept. Elaboration also attempts to tackle ambiguities, unknowns, and inconsistencies in the higher-level description.

Example: The conceptual idea “Manipulate numbers to get as close as you can to the target” was elaborated into the following ideas: “Add the first three numbers and then add the next three and then check each three.” Figure 2b illustrates the structural representation of the elaboration of the first sentence by the second sentence. Note that the second sentence details the meaning of the verb (action) “manipulate”. The figure also presents the connection of the two ideas to the problem description, e.g., getting as close as possible to the target in the first sentence and adding triplets in the second. The elaboration includes the sequencing of the three verbs (actions). Additional prompting must address the ambiguities in the second sentence, like finding the concept to which “three” refers to.

Figure 2a illustrates the proposed elaboration method in which prompts are adaptively generated to prompt an LLM to provide the information needed to detail a higher-level idea. It is assumed that an elaboration of a group of words that can include concepts (e.g., nouns and prepositions), verbs (relations), and features is achieved by creating one or more ideas (sentences) that use action or identity verbs [55] to offer a more precise description of the elaborated group. The outcome (meaning) of an elaboration expresses the elaborated group. Each generated prompt has two parts: the request to perform an activity related to elaboration (i.e., finding the significant words, identifying the relations linking words, etc.) and the immediate context of the prompt, which summarizes the immediately preceding sentences and the rest of the sentence to which the elaborated group belongs. The information for prompt generation comes from the KG representation for the short-term and episodic memories. The produced ideas are then linked together to reflect property-based and relation-based combinations. Figure 2b shows the elaboration process of the above example.

Elaboration can be theoretically expressed as follows. Frieder et al. propose that a transformer τ with L levels is described by the following mathematical relation [56]:

$$\tau = (Id + M_L^* \circ N_{M,L}^*) \circ (Id + A_L^* \circ N_{A,L}^*) \circ \dots \circ (Id + M_1^* \circ N_{M,1}^*) \circ (Id + A_1^* \circ N_{A,1}^*) \quad (1)$$

Variable Id is the identity mapping (skip), parameters A_i^* are self-attention maps, parameters M_i^* are feedforward multi-layer perceptrons, and parameters $N_{M,i}^*$ and $N_{A,i}^*$ are normalization layers.

Equation (1) shows that a transformer distributes meaning across compositions of attention and feed-forward blocks. The CA mirrors this compositional structure at the symbolic level: rather than recovering the learned weights, it operates on an editable graph whose dependency edges play a functionally analogous role to the attention products in (1). When a group of words W triggers an elaboration request, the controller retrieves the minimal subgraph of $\mathcal{G}_{\text{task}}$ whose neighborhood corresponds to the relevant compositions.

Elaborations act as symbolic counterparts to the latent products $M_i^* \circ N_{M,i}^*$ and $A_j^* \circ N_{A,j}^*$, accepted only when the induced subgraph preserves predicate–argument roles. This parallel serves as a design motivation for the graph-update mechanism; establishing a formal equivalence between the two representations remains an open theoretical question.

2. Create a conceptual representation of the related ideas. To extract conceptual structure from speech dialog, the proposed techniques build on the sentence-level KG representation described earlier. Each sentence is first transformed into a structural graph, where words are nodes and intra-sentence syntactic or semantic relations are edges, like shown in Figures 1a and 2b. This transformation preserves the compositional semantics and role information necessary to ground any problem-solving strategy.

Our work explored four KG-based approaches to organize and interpret the dialog flow to create a conceptual representation:

1. **Hard membership:** Sentence-level KGs are added only to the previous cluster if semantically similar; otherwise, a new cluster is created. Clusters form a linear or branching structure with one-to-one connections, enforcing non-overlapping progression.
2. **Soft membership:** Sentence-level KGs can belong to multiple clusters based on semantic similarity. Clusters are interconnected with multiple others, capturing overlapping ideas and parallel reasoning paths.
3. **Sentence-to-sentence hooks:** The method skips clustering by directly linking sentence-level KGs through semantic relations such as co-reference or logical continuation, enabling fine-grained tracking of idea flow.
4. **Pruning:** The method refines the soft membership graph by applying automated metrics (e.g., silhouette score) to remove weak links and merge redundant clusters, thus simplifying the structure while preserving key semantics.

This layered graph representation supports identifying and sequencing the five problem-solving strategies by tracing how concepts are introduced, elaborated, combined, or abstracted across the dialog. It bridges the problem description and solution space by structurally capturing how knowledge evolves in collaborative settings.

3. Create new ideas through combinations. Concept combination is the mechanism through which new ideas are produced by relating fragments or entire ideas to emulate property-based and relation-based combinations in cognitive psychology. Mathematically, combinations are described by the following relation:

$$C(C_k(w_a^{(1)}, w_b^{(1)}, w_x^{(2)}), C_m(w_p^{(1)}, w_q^{(1)}, w_l^{(2)}, w_{ll}^{(2)}), \dots C_p(\dots)) \quad (2)$$

where C and C_i describe a concept combination that involves words $w^{(1)}$ and $w^{(2)}$ from two sentences. The relation expresses a two-layered combination process, in which new meanings are created by combining (C_i) groups of words followed by combining (C) of the meanings of the groups into an overall meaning of the new idea. Property-based combinations transfer features from one concept to another concept (e.g., a concept adds numbers and another concept sorts strings, the new concept sorts numbers), or establish new relations (through specific verbs) that can represent causality (i.e., using action verbs), identity and structuring (e.g., through verbs like “is” and “has”), support (enabling) and co-occurrence (association).

As a first step towards implementing more comprehensive concept combination methods, new methods should consider different parameters, such as the roles that concepts can have in a combination, their sensitivity with respect to the outcome of the combination, and different ways of creating causality, co-occurrences, and enabling relations between concepts.

5. Empirical Validation

5.1. Experimental Setup

To empirically assess the proposed LLM-based Cognitive Architecture (CA), we address the three research questions stated in Section 1.

To orient the reader: results for **RQ1** appear in Table 2 (Adaptive Prompting); **RQ2** is addressed by Table 3 (Strategy Utilization). Because strategy scheduling depends on the conceptual-representation scaffold (Section 4, “Create a conceptual representation of the related ideas”), we additionally report an ablation of those techniques in Table 4 to contextualize **RQ2**, and **RQ3** appears in Table 5 (Concept Combination Impact).

Table 2. Adaptive vs. Standard Prompting: Quantitative Improvements (averaged across nine CCS scenarios; three per difficulty level).

Method	Avg. Semantic Coherence	Completeness (%)	Dialog Turns
Standard CoT	0.76	68	9.2
Adaptive Prompting (ours)	0.83	81	7.5
Improvement	+9.2%	+19.1%	−18.5%

Table 3. Cognitive Strategy Usage Across Task Difficulties (averaged across nine CCS scenarios; three per difficulty level).

Task Difficulty	S ₁	S ₂	S ₃	S ₄	S ₅
Easy	48%	14%	10%	13%	15%
Medium	35%	21%	18%	16%	10%
Hard	26%	25%	24%	17%	8%

Experimental Setting.

The empirical validation was conducted at the Department of Electrical and Computer Engineering, Stony Brook University, Stony Brook, NY, USA. Participants were undergraduate students enrolled in a programming course. Groups of three students each worked together in a supervised laboratory environment to collaboratively solve programming tasks. All participants had prior exposure to programming at the undergraduate level; no specific selection criteria beyond course enrollment were applied. Sessions were audio-recorded and later transcribed to form the dialog corpus.

(a) Datasets and Tasks.

We employ the **Structured Code Synthesis** (CCS) dataset: 30 undergraduate dialogs where groups of three students collaboratively solved the same programming task. We analyse **nine representative scenarios** (three per difficulty level: Easy, Medium, Hard) selected based on ambiguity and uncertainty in their discussions. Due to participant privacy constraints, the dataset is not publicly released; anonymized excerpts supporting reproducibility are available from the corresponding author upon reasonable request.

(b) Implementation.

Dialog transcripts are pre-processed through a two-stage pipeline. First, dialogs are tokenized and dependency-parsed with spaCy 3.7; the resulting predicate–argument structures directly populate the nodes and typed edges of $\mathcal{G}_{\text{task}}$, while part-of-speech tags distinguish action verbs from identity verbs. Second, close-world words (e.g., conjunctions such as and/or that signal equivalence and alternative roles) are classified by the stacked Multi-Layered Perceptron described in [9], which also identifies the hierarchical sentence structure and word-role assignments. The CA controller (Algorithm 1) then assembles

prompts from this structured representation and submits them to GPT-5; parser confidence scores are used by the controller during the validation step to flag low-confidence edges for re-prompting. We instantiate four alternative KG maintenance policies (Hard, Soft, Hooks, Pruning) to study their impact on strategy scheduling.

Table 4. Conceptual-representation technique-wise performance (mean $\pm$ SD) across dialog complexity.

Dialogue Complexity	CU	SA	RR	SS	ISW	SC	Avg
Hard Membership							
Easy	7.1 $\pm$ 0.3	6.2 $\pm$ 0.5	5.4 $\pm$ 0.7	8.0 $\pm$ 0.6	6.5 $\pm$ 0.4	6.3 $\pm$ 0.6	6.6 $\pm$ 0.8
Medium	6.0 $\pm$ 0.4	5.5 $\pm$ 0.6	7.2 $\pm$ 0.5	6.4 $\pm$ 0.3	7.0 $\pm$ 0.7	7.1 $\pm$ 0.4	6.5 $\pm$ 0.7
Hard	5.2 $\pm$ 0.6	4.1 $\pm$ 0.4	9.0 $\pm$ 0.8	4.2 $\pm$ 0.5	8.1 $\pm$ 0.3	7.0 $\pm$ 0.6	6.3 $\pm$ 0.9
Soft Membership							
Easy	9.2 $\pm$ 0.2	9.0 $\pm$ 0.3	9.1 $\pm$ 0.2	9.0 $\pm$ 0.4	3.5 $\pm$ 0.7	4.0 $\pm$ 0.5	7.3 $\pm$ 0.8
Medium	8.0 $\pm$ 0.3	8.2 $\pm$ 0.2	8.1 $\pm$ 0.4	8.0 $\pm$ 0.3	5.0 $\pm$ 0.5	5.2 $\pm$ 0.6	7.1 $\pm$ 0.6
Hard	7.1 $\pm$ 0.5	7.0 $\pm$ 0.4	7.2 $\pm$ 0.5	7.1 $\pm$ 0.4	6.0 $\pm$ 0.6	4.3 $\pm$ 0.7	6.4 $\pm$ 0.7
Sentence-to-Sentence Hooks							
Easy	3.2 $\pm$ 0.4	3.0 $\pm$ 0.3	4.1 $\pm$ 0.5	5.0 $\pm$ 0.4	4.2 $\pm$ 0.3	4.0 $\pm$ 0.5	3.9 $\pm$ 0.6
Medium	2.1 $\pm$ 0.5	2.0 $\pm$ 0.4	6.1 $\pm$ 0.7	4.3 $\pm$ 0.3	6.0 $\pm$ 0.6	4.1 $\pm$ 0.4	4.1 $\pm$ 0.8
Hard	1.0 $\pm$ 0.3	1.2 $\pm$ 0.4	8.0 $\pm$ 0.8	3.2 $\pm$ 0.3	7.1 $\pm$ 0.5	3.0 $\pm$ 0.4	3.9 $\pm$ 0.9
Pruning (Post Soft Membership)							
Easy	8.1 $\pm$ 0.4	8.0 $\pm$ 0.3	7.2 $\pm$ 0.5	8.0 $\pm$ 0.4	6.4 $\pm$ 0.4	6.1 $\pm$ 0.3	7.3 $\pm$ 0.6
Medium	7.1 $\pm$ 0.5	7.0 $\pm$ 0.4	7.0 $\pm$ 0.4	7.0 $\pm$ 0.3	7.1 $\pm$ 0.5	6.0 $\pm$ 0.3	6.9 $\pm$ 0.6
Hard	6.2 $\pm$ 0.4	6.1 $\pm$ 0.4	8.0 $\pm$ 0.6	6.1 $\pm$ 0.4	7.2 $\pm$ 0.5	6.0 $\pm$ 0.3	6.6 $\pm$ 0.7

Each cell in this table reports the mean ($\pm$ standard deviation) performance computed manually across the nine CCS scenarios, aggregated over three dialogue complexity levels (Easy, Medium, and Hard) for each technique-metric pair. Specifically, for every technique and complexity level, scores were first averaged across all scenario instances within that level (three per category), and then averaged across the nine total scenarios. The resulting value thus reflects the adjudicated mean performance of the LLM over repeated trials, while the accompanying standard deviation captures the variance observed among the scenarios for that configuration. Abbreviations: CU (concept understanding), SA (semantic alignment), RR (reasoning recall), SS (sequence score), ISW (integration with specification and working plan), SC (solution completeness).

Table 5. Effect of explicit concept combinations (averaged across nine CCS scenarios; three per difficulty level). “Baseline” here denotes adaptive prompting *without* the combination module, not the CoT baseline of Table 2.

Metric	Baseline	+Combination	Δ
Distinct-3 (%)	12.8	25.2	+12.4
Creativity (1–5)	3.1	3.8	+0.7
Semantic coherence	0.74	0.81	+0.07

(c) Metrics.

We report (i) semantic coherence, computed as the cosine similarity between predicted and ground-truth intent embeddings; (ii) solution completeness, measured against instructor rubrics (0–100); (iii) dialog turns required to reach an acceptable plan; (iv) novelty, using distinct n -gram ratios rather than BLEU to avoid unstable percentages on short texts; and (v) human-rated creativity on a five-point Likert scale by two annotators ($\kappa = 0.71$).

5.2. Results and Discussion

(a) Adaptive Prompting (RQ1):

Table 2 summarizes the comparison against a Chain-of-Thought (CoT) baseline that reuses our prompts but without adaptive goal selection. Adaptive prompting yields higher

semantic coherence (+9.2%) and solution completeness (+19.1%) while reducing dialog turns by 18.5%, showing that the controller accelerates convergence without sacrificing alignment to instructor rubrics. All values are averaged across nine CCS scenarios (three per difficulty level).

(b) Strategy Utilization (RQ2):

Table 3 quantitatively highlights the CA's adaptive strategy deployment across varying difficulties. Hard dialogs reduce reliance on pure elaboration (S_1) to 26% while elevating combination strategies (S_2/S_3) to the mid-20s, indicating that the controller allocates more combinational reasoning as ambiguity rises. Specifically: S_1 decreases 48%→35%→26%; S_2 increases 14%→21%→25%; S_3 increases 10%→18%→24%; S_4 is roughly stable 13%→16%→17%; S_5 drops 15%→10%→8%. Percentages reflect averages across nine CCS scenarios (three per difficulty level).

Metrics: Distinct-3 is the ratio of unique 3-g to total 3-g; Creativity is a human Likert rating by two annotators ($\kappa = 0.71$); Semantic coherence is cosine similarity between predicted and ground-truth intent embeddings (see (c) Metrics).

(c) Concept Combination Impact (RQ3):

Table 5 quantifies the benefits of explicit concept combinations. Distinct n -gram novelty rises by 12.4 points (from 12.8% to 25.2%, $\approx +97\%$ relative), human-rated creativity improves by 0.7 on the five-point scale, and semantic coherence increases by 0.07 (from 0.74 to 0.81, $\approx +9.5\%$ relative), indicating that feature-level blending yields more diverse yet still aligned proposals. *Implications for the CA.* These gains empirically support the CA's combination module (Section 4; Figure 2a, item 3). They show that property-based transfers and relation-inducing links (Section 3, Equation (2)) raise generative diversity without sacrificing alignment. In tandem with the difficulty-wise rise of S_2/S_3 usage (Table 3), Table 5 validates the controller's choice to allocate more combination steps as ambiguity increases. The observed coherence gain is consistent with KG-grounded validation in Algorithm 1, which filters incoherent edges before graph updates.

5.3. Empirical Evidence for Conceptual Representation

We ablate the four KG-based conceptual-representation techniques introduced in Section 4 ("Create a conceptual representation of the related ideas") on the CCS corpus (30 collaborative programming dialogs across nine scenarios). Because these techniques gate downstream strategy scheduling, their performance across dialogue complexity contextualizes the RQ2 results.

(d) Observations (Table 4):

- **Soft Membership (Easy/Medium):** Highest CU/SA/RR (e.g., Easy CU = 9.2, SA = 9.0, RR = 9.1), but lowest ISW/SC (Easy ISW = 3.5, SC = 4.0).
- **Hard Membership (Hard):** Strongest RR/ISW/SC on Hard (RR = 9.0, ISW = 8.1, SC = 7.0), with lower CU/SA (5.2/4.1).
- **Sentence-to-Sentence Hooks:** RR scales with difficulty (4.1→6.1→8.0), but CU/SA remain low (Easy 3.2/3.0).
- **Pruning (post-Soft):** Best balance: on Easy, ISW improves over Soft (3.5→6.4) and SC (4.0→6.1) while keeping CU/SA high (8.1/8.0); on Hard, achieves RR/ISW/SC of 8.0/6.1/6.0 with higher CU/SA than Hooks.

Takeaway: Use Soft to drive early elaboration, prune before integration, and shift toward Hard-style partitioning as ambiguity rises. These patterns also suggest how the scaffold and controller operate in practice. Overlapping neighborhoods (Soft) appear to aid elaboration on easier dialogs (Figure 2a, item 1), whereas the drop in ISW/SC

highlights the need for validation and pruning before accepting graph updates. Pruning improves integration and completeness, consistent with connectivity and coverage checks in Algorithm 1. On harder dialogs, tighter, non-overlapping partitions (Hard) seem to better support downstream planning under ambiguity, aligning with the observed rise in combination-oriented steps (S_2/S_3). Sentence-to-sentence hooks strengthen RR and help trigger combinations (Figure 2a, item 3; Equation (2)) but benefit from a global scaffold to preserve conceptual coherence. Overall, the ablation connects conceptual-representation choices to strategy scheduling across task complexity (RQ2).

5.4. Scope and Future Directions

The present study validates the CA on a controlled, IRB-approved corpus of 30 collaborative programming dialogues with ground-truth instructor rubrics. This design choice enables rigorous, rubric-based measurement of strategy scheduling and concept combination but naturally bounds the current experimental scope. Several directions can extend these results. First, evaluation on larger corpora, including asynchronous discussions from open-source platforms, would further demonstrate scalability. Second, the CoT baseline was chosen to isolate the effect of adaptive goal selection; additional comparisons with modern agentic frameworks (e.g., AutoGen, ReAct-based pipelines) would help characterise the specific contribution of the KG scaffold. Third, while the theoretical model and the CA design are domain-agnostic, cross-domain studies (e.g., collaborative design, scientific inquiry) are a natural next step. Finally, replacing the manually tuned thresholds (τ_{add} , τ_{rev}) with learned parameters is a promising avenue for improving adaptability.

6. Summary of Research Needs

This paper argues that new ML research should address four main needs: bridging the stochastic vs. symbolic modeling gap, idea elaboration, combining concepts, and understanding vs. associative learning. Tackling these needs requires new research on the following topics: a theoretical model to integrate stochastic and symbolic representations, a mathematical way of expressing idea elaboration, a method to combine concepts beyond the traditional capabilities of functions and function composition, and expanding ML modeling with capabilities of producing internal representations (similar to mental representations) to support understanding. While LLMs are a significant breakthrough in the general-purpose modeling of knowledge, new research is needed to leverage their generality towards effectively addressing the generative needs of a specific problem.

Our empirical study indicates that editable KG scaffolds provide the interpretable control layer needed for scalable collaboration, complementing purely associative approaches. Such hybrid architectures retain transparency while remaining responsive to live team intent, suggesting a practical path toward the four outlined needs.

Author Contributions: Conceptualization, H.S., G.V. and A.D.; methodology, H.S. and G.V.; software, H.S.; validation, H.S., G.V. and A.D.; formal analysis, A.D.; investigation, H.S.; resources, A.D.; data curation, G.V.; writing—original draft preparation, H.S.; writing—review and editing, A.D.; visualization, H.S.; supervision, A.D.; project administration, A.D.; funding acquisition, A.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: The study was conducted in accordance with the Declaration of Helsinki and approved by the Institutional Review Board of Stony Brook University.

Informed Consent Statement: Informed consent was obtained from all participants involved in the study.

Data Availability Statement: The Structured Code Synthesis (CCS) dataset used in this study is not publicly available due to participant privacy constraints. Anonymized excerpts supporting the reproducibility of the reported results are available from the corresponding author upon reasonable request.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Simonton, D.K. Creative Thought as Blind-Variation and Selective-Retention: Combinatorial Models of Exceptional Creativity. *Phys. Life Rev.* **2010**, *7*, 156–179. [CrossRef]
2. Doboli, A.; Umbarkar, A. The Role of Precedents in Increasing Creativity During Iterative Design of Electronic Embedded Systems. *Des. Stud.* **2014**, *35*, 298–326. [CrossRef]
3. Li, H.; Liu, X.; Jiao, F.; Doboli, A.; Doboli, S. InnovA: A Cognitive Architecture for Computational Innovation Through Robust Divergence and Its Application for Analog Circuit Design. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2018**, *37*, 1943–1956. [CrossRef]
4. Wang, R.; Lehman, J.; Rawal, A.; Zhi, J.; Li, Y.; Clune, J.; Stanley, K. Enhanced POET: Open-Ended Reinforcement Learning through Unbounded Invention of Learning Challenges and Their Solutions. In Proceedings of the International Conference on Machine Learning, Online, 13–18 July 2020; pp. 9940–9951.
5. Ainsworth, S.; Prain, V.; Tytler, R. Drawing to Learn in Science. *Science* **2011**, *333*, 1096–1097. [CrossRef]
6. Christensen, B.; Schunn, C. The Relationship of Analogical Distance to Analogical Function and Preinventive Structure: The Case of Engineering Design. *Mem. Cogn.* **2007**, *35*, 29–38. [CrossRef]
7. Hwang, W.; Chen, N.; Dung, J.; Yang, Y. Multiple Representation Skills and Creativity Effects on Mathematical Problem Solving Using a Multimedia Whiteboard System. *Educ. Technol. Soc.* **2007**, *10*, 191–212.
8. Fiore, S.; Rosen, M.; Smith-Jentsch, K.; Salas, E.; Letsky, M.; Warner, N. Toward an Understanding of Macrocognition in Teams: Predicting Processes in Complex Collaborative Contexts. *Hum. Factors* **2010**, *52*, 203–224. [CrossRef]
9. Villuri, G.; Shaik, H.; Doboli, A.; Doboli, S. A Stacked Multi-Layered Perceptron—LLM Model for Extracting the Relations in Textual Descriptions. In Proceedings of the 2025 IEEE Symposium on Computational Intelligence in Natural Language Processing and Social Media (CI-NLPSoMe Companion), Trondheim, Norway, 17–20 March 2025; pp. 1–5.
10. Bishop, C.; Bishop, H. *Deep Learning: Foundations and Concepts*; Springer: Cham, Switzerland, 2023.
11. Cohen, B.; Murphy, G. Models of Concepts. *Cogn. Sci.* **1984**, *8*, 27–58. [CrossRef]
12. Barsalou, L.W. Grounded Cognition. *Annu. Rev. Psychol.* **2008**, *59*, 617–645. [CrossRef]
13. Doboli, A.; Vemuri, R. Behavioral Modeling for High-Level Synthesis of Analog and Mixed-Signal Systems from VHDL-AMS. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2003**, *22*, 1504–1520. [CrossRef]
14. Kruiskamp, W.; Leenaerts, D. DARWIN: CMOS Opamp Synthesis by Means of a Genetic Algorithm. In Proceedings of the 32nd Annual ACM/IEEE Design Automation Conference, San Francisco, CA, USA, 12–16 June 1995; pp. 433–438.
15. McConaghy, T.; Palmers, P.; Peng, G.; Steyaert, M.; Gielen, G. *Variation-Aware Analog Structural Synthesis*; Springer: Dordrecht, The Netherlands, 2009.
16. Estes, Z.; Ward, T. The Emergence of Novel Attributes in Concept Modification. *Creat. Res. J.* **2002**, *14*, 149–156. [CrossRef]
17. Wisniewski, E. When Concepts Combine. *Psychon. Bull. Rev.* **1997**, *4*, 167–183. [CrossRef]
18. Zahedi, M.A.; Gholamrezaei, N.; Doboli, A. How Deep Is Your Art: An Experimental Study on the Limits of Artistic Understanding in a Single-Task, Single-Modality Neural Network. *PLoS ONE* **2024**, *19*, e0305943.
19. Manhaeve, R.; Dumancic, S.; Kimmig, A.; Demeester, T.; De Raedt, L. DeepProbLog: Neural Probabilistic Logic Programming. *Adv. Neural Inf. Process. Syst.* **2018**, *31*, 3753–3763. [CrossRef]
20. Parmar, M.; Patel, N.; Varshney, N.; Nakamura, M.; Luo, M.; Mashetty, S.; Mitra, A.; Baral, C. LogicBench: Towards Systematic Evaluation of Logical Reasoning Ability of Large Language Models. *arXiv* **2024**, arXiv:2404.15522.
21. De Smet, L.; Venturato, G.; De Raedt, L.; Marra, G. Relational Neurosymbolic Markov Models. *arXiv* **2024**, arXiv:2412.13023. [CrossRef]
22. Feng, J.; Xu, R.; Hao, J.; Sharma, H.; Shen, Y.; Zhao, D.; Chen, W. Language Models Can Be Deductive Solvers. In *Findings of the Association for Computational Linguistics: NAACL 2024*; Association for Computational Linguistics: San Diego, CA, USA, 2024; pp. 4026–4042.
23. Choi, Y.; Vergari, A.; den Broeck, G.V. Probabilistic Circuits: A Unifying Framework for Tractable Probabilistic Models. Technical Report. 2020. Available online: <https://yoojungchoi.github.io/files/ProbCirc20.pdf> (accessed on 4 March 2026).
24. Liao, H.; He, S.; Xu, Y.; Zhang, Y.; Liu, K.; Zhao, J. Neural-Symbolic Collaborative Distillation: Advancing Small Language Models for Complex Reasoning Tasks. In Proceedings of the AAAI Conference on Artificial Intelligence, Philadelphia, PA, USA, 25 February–4 March 2025.

25. Saha, S.; Yadav, P.; Bansal, M. *multiPROver*: Generating Multiple Proofs for Improved Interpretability in Rule Reasoning. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL 2021), Online, 6–11 June 2021; pp. 3662–3677.
26. Franz Inc. AllegroGraph v8: Neuro-Symbolic AI Platform Integrating Knowledge Graphs and LLMs. White Paper. 2024. Available online: <https://franz.com/agraph/> (accessed on 4 March 2026).
27. Barrault, L.; Duquenne, P.-A.; Elbayad, M.; Kozhevnikov, A.; Alastruey, B.; Andrews, P.; Coria, M.; Couairon, G.; Costa-jussà, M.R.; Dale, D.; et al. Large Concept Models: Language Modeling in a Sentence Representation Space. *arXiv* **2024**, arXiv:2412.08821. [CrossRef]
28. Trinh, T.H.; Wu, Y.; Le, Q.V.; He, H.; Luong, T. Solving Olympiad Geometry without Human Demonstrations. *Nature* **2024**, *625*, 476–482. [CrossRef]
29. Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Ichter, B.; Xia, F.; Chi, E.; Le, Q.; Zhou, D. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. *arXiv* **2022**, arXiv:2201.11903.
30. Yao, S.; Zhao, J.; Yu, D.; Du, N.; Shafran, I.; Narasimhan, K.; Cao, Y. ReAct: Synergizing Reasoning and Acting in Language Models. *arXiv* **2022**, arXiv:2210.03629.
31. Wang, L.; Xu, W.; Lan, Y.; Hu, Z.; Lan, Y.; Lee, R.K.-W.; Lim, E.-P. Plan-and-Solve Prompting: Improving Zero-Shot Chain-of-Thought Reasoning by Large Language Models. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics, Toronto, ON, Canada, 9–14 July 2023; pp. 2609–2634.
32. Yao, S.; Yu, D.; Zhao, J.; Shafran, I.; Griffiths, T.L.; Cao, Y.; Narasimhan, K. Tree of Thoughts: Deliberate Problem Solving with Large Language Models. *Adv. Neural Inf. Process. Syst.* **2023**, *36*, 11809–11822.
33. Wang, X.; Wei, J.; Schuurmans, D.; Le, Q.; Chi, E.; Narang, S.; Chowdhery, A.; Zhou, D. Self-Consistency Improves Chain of Thought Reasoning in Language Models. *arXiv* **2022**, arXiv:2203.11171.
34. Zhang, Z.; Zhang, A.; Li, M.; Smola, A. Automatic Chain of Thought Prompting in Large Language Models. *arXiv* **2022**, arXiv:2210.03493. [CrossRef]
35. Wang, G.; Xie, Y.; Jiang, Y.; Mandlekar, A.; Xiao, C.; Zhu, Y.; Fan, L.; Anandkumar, A. Voyager: An Open-Ended Embodied Agent with Large Language Models. *arXiv* **2023**, arXiv:2305.16291. [CrossRef]
36. Qian, C.; Liu, W.; Liu, H.; Chen, N.; Dang, Y.; Li, J.; Yang, C.; Chen, W.; Su, Y.; Cong, X.; et al. ChatDev: Communicative Agents for Software Development. *arXiv* **2023**, arXiv:2307.07924.
37. Yang, J.; Jimenez, C.E.; Wettig, A.; Lieret, K.; Yao, S.; Narasimhan, K.; Press, O. SWE-Agent: Agent-Computer Interfaces Enable Automated Software Engineering. *Adv. Neural Inf. Process. Syst.* **2024**, *37*, 50528–50652.
38. Wu, Q.; Bansal, G.; Zhang, J.; Wu, Y.; Li, B.; Zhu, E.; Jiang, L.; Zhang, X.; Zhang, S.; Liu, J.; et al. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation. *arXiv* **2023**, arXiv:2308.08155.
39. Chang, E.Y.; Geng, L. SagaLLM: Context Management, Validation, and Transaction Guarantees for Multi-Agent LLM Planning. *arXiv* **2025**, arXiv:2503.11951. [CrossRef]
40. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient Estimation of Word Representations in Vector Space. *arXiv* **2013**, arXiv:1301.3781. [CrossRef]
41. Pennington, J.; Socher, R.; Manning, C.D. GloVe: Global Vectors for Word Representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014; pp. 1532–1543.
42. Nickel, M.; Tresp, V.; Krieger, H.P. A Three-Way Model for Collective Learning on Multi-Relational Data. In Proceedings of the 28th International Conference on Machine Learning, Bellevue, WA, USA, 28 June–2 July 2011; pp. 809–816.
43. Socher, R.; Huval, B.; Manning, C.D.; Ng, A.Y. Semantic Compositionality through Recursive Matrix–Vector Spaces. In Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, Jeju Island, Republic of Korea, 12–14 July 2012; pp. 1201–1211.
44. Wang, Z.; Gui, L.; Negrea, J.; Veitch, V. Concept Algebra for (Score-Based) Text-Controlled Generative Models. *arXiv* **2023**, arXiv:2302.03693.
45. Su, J.; Liu, N.; Wang, Y.; Tenenbaum, J.B.; Du, Y. Compositional Image Decomposition with Diffusion Models. *arXiv* **2024**, arXiv:2406.19298. [CrossRef]
46. Johnson, J.; Hariharan, B.; Maaten, L.v.; Fei-Fei, L.; Zitnick, C.L.; Girshick, R. CLEVR: A Diagnostic Dataset for Compositional Language and Elementary Visual Reasoning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 1988–1997.
47. Asai, A.; Wu, Z.; Wang, Y.; Sil, A.; Hajishirzi, H. Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection. *arXiv* **2023**, arXiv:2310.11511. [CrossRef]
48. Gutiérrez, B.J.; Shu, Y.; Gu, Y.; Yasunaga, M.; Su, Y. HippoRAG: Neurobiologically Inspired Long-Term Memory for Large Language Models. *arXiv* **2024**, arXiv:2405.14831.

49. Hu, M.; Zong, L.; Wang, H.; Zhou, J.; Li, J.; Gao, Y.; Wong, K.; Li, Y.; King, I. SeRTS: Self-Rewarding Tree Search for Biomedical Retrieval-Augmented Generation. In *Findings of the Association for Computational Linguistics: EMNLP 2024*; Association for Computational Linguistics: Miami, FL, USA, 2024; pp. 1321–1335.
50. Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C.L.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; et al. Training Language Models to Follow Instructions with Human Feedback. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 27730–27744.
51. Lee, H.; Phatale, S.; Mansoor, H.; Mesnard, T.; Ferret, J.; Lu, K.; Bishop, C.; Hall, E.; Carbune, V.; Rastogi, A.; et al. RLAIIF vs. RLHF: Scaling Reinforcement Learning from Human Feedback with AI Feedback. *arXiv* **2023**, arXiv:2309.00267.
52. Rafailov, R.; Sharma, A.; Mitchell, E.; Ermon, S.; Manning, C.D.; Finn, C. Direct Preference Optimization: Your Language Model Is Secretly a Reward Model. *Adv. Neural Inf. Process. Syst.* **2023**, *36*, 53728–53741.
53. Chae, H.; Kim, N.; Ong, K.T.; Gwak, M.; Song, G.; Kim, J.; Kim, S.; Lee, D.; Yeo, J. Web Agents with World Models: Learning and Leveraging Environment Dynamics in Web Navigation. *arXiv* **2024**, arXiv:2410.13232. [[CrossRef](#)]
54. Zhang, A.; Nguyen, K.; Tuyls, J.; Lin, A.; Narasimhan, K. Language-Guided World Models: A Model-Based Approach to AI Control. *arXiv* **2024**, arXiv:2402.01695.
55. Fauconnier, G.; Turner, M. *The Way We Think. Conceptual Blending and the Mind's Hidden Complexities*; Basic Books: New York, NY, USA, 2002.
56. Frieder, S.; Berner, J.; Petersen, P.; Lukasiewicz, T. Large Language Models for Mathematicians. *arXiv* **2024**, arXiv:2312.04556. [[CrossRef](#)]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.