Proceedings of the Design Society, Volume 5: ICED25
https://doi.org/10.1017/pds.2025.10239

Leveraging large language models for enabling design by analogy: a computational framework

Rohin Joshi[1], Ruhi Mitra[1], Vijayalaxmi Sahadevan[1], Kane Borg[2], Vishal Singh[1], Bilal Muhammed[3], Soban Babu Beemaraj[3] and Amol Joshi[3]

[1] Indian Institute of Science Bangalore, India, [2] Aalto University, Finland, [3] TCS Research & Innovation, Tata Consultancy Services, Pune, Maharashtra, India

✉ svijaya16@gmail.com

ABSTRACT: Design by Analogy (DbA) is a powerful method for fostering innovation by transferring knowledge from a source domain to solve problems in a target domain. However, traditional DbA approaches face significant challenges, including resource-intensive database management, linguistic and representational differences across domains, and the complexity of access and mapping processes. These limitations hinder scalability and efficiency, particularly for cross-domain analogies. Recent advancements in Artificial Intelligence (AI), especially Large Language Models (LLMs), offer promising solutions by facilitating efficient knowledge retrieval, bridging linguistic gaps, and enhancing semantic reasoning. This paper explores the potential of AI technologies to address these challenges, proposing a framework for analogical reasoning.

KEYWORDS: Design by Analogy, Artificial intelligence, Knowledge management

1. Introduction

Analogy is a powerful strategy for creative problem-solving, enabling designers to draw insights from similar or cross-domain contexts (Holyoak & Thagard, 1996; Singh, Casakin, et al., 2015) Effective Design by Analogy (DbA) requires representing source and target knowledge in a unified framework, typically stored in databases for systematic retrieval and mapping. However, maintaining these databases is resource-intensive, especially for cross-domain analogies, which face linguistic and representational challenges.

DbA relies on **access** (retrieving relevant analogies) and **mapping** (aligning source and target elements), both of which are cognitively and computationally demanding. Recent advances in Artificial Intelligence (AI), particularly Large Language Models (LLMs), enhance these processes by efficiently storing, retrieving, and reasoning about knowledge across domains.

This paper proposes a systematic DbA pipeline integrating LLMs with graph algorithms to automate and scale analogy-driven design. The framework follows five stages: **Retrieval**, **Mapping**, **Transfer**, **Evaluation**, and **Storage** (Ball & Christensen, 2022), leveraging LLMs for semantic understanding and graph algorithms for structural organization. This approach enhances efficiency, scalability, and cross-domain applicability, addressing limitations in traditional DbA methods.

2. Literature review

Given its importance in creative problem-solving, numerous DbA approaches have emerged, including biomimetic design and analogical reasoning in engineering and design. These methods can be categorized based on their methodological principles and focus areas.

Natural language processing (NLP) and text mining techniques are often used to bridge terminological gaps between biological and engineering domains. For instance, (Chiu & Shu, 2007) proposed a bridging method using NLP to uncover less-obvious connections between engineering and biological terminology. Similarly, (Verhaegen et al., 2011) applied word co-occurrence and principal component analysis (PCA) to analyze patent data for identifying DbA candidates.(Vandevenne et al., 2016) developed SEABIRD, which maps technical systems described in patents to biological systems referenced in academic literature.

Function-based methodologies represent another key category, focusing on the functional characteristics of designs. (Stone & Wood, 1999) introduced a functional basis framework for representing design. (Fu et al., 2015) proposed a patent-based analogy search using functional vector approaches, while (Briana et al., 2015) presented tools such as D-APPS and DRACULA, which integrate functional models with resources like WordNet and the AskNature repository. (Sanaei et al., 2017) devised a text-based system leveraging engineering ontologies and hierarchical function representations to retrieve design analogies.

In addition, computational models have been developed to facilitate biomimetic design and analogical reasoning. For example, (Grace et al., 2015) introduced Idiom, a computational model for analogical mapping that reinterprets object representations. (Oriakhi et al., 2011) created the WordTree method and its associated tool, WordTree Express (WTE), which visually represent word relationships based on functional design principles. Tools like (Vattam et al., 2011) and VISION (Song et al., 2020) provide innovative approaches to structure-behavior-function modeling and visual interaction for analogical inspiration, respectively.

Model-based analogy approaches offer deeper insights by capturing structural and functional similarities. (Goel & Bhatta, 2004) explored model-based analogy (MBA), which transfers generic teleological mechanisms (GTMs) between contexts. Other notable contributions include (Goel et al., 1997), which uses functional basis models for design modification and verification, and IDEAL (Bhatta &Goel, 1996), which extracts generic teleological mechanisms for analogical mapping.

Case-based reasoning (CBR), rooted in analogical reasoning, is another influential method (Hybs and Gero, 1992). CBR relies on previously encountered cases, using similarity measures to retrieve relevant instances that inform new problem-solving scenarios (Perner, 2014). The CBR process includes four stages—retrieve, reuse, revise, and retain (Aamodt and Plaza, 1994)—to emulate human reasoning. Applications of CBR span various design domains, including architectural design (Mubarak, 2004) and mechanical device development (Qin and Regli, 2003).

Overall, most DbA-inspired approaches focus on ideation and solution recommendation within predefined problem contexts. Earlier, DbA approaches faced significant challenges in managing cross-domain analogies due to linguistic and representational differences, requiring extensive manual effort or rigid databases for analogy retrieval and mapping. These methods often struggled with scalability and flexibility, limiting their applicability in diverse contexts. In contrast, using LLMs enables seamless integration of linguistic, contextual, and semantic reasoning, offering enhanced adaptability and efficiency in retrieving and mapping analogies across varied domains.

3. Methodology

3.1. Overall architecture

The proposed framework implements a systematic pipeline for automating and scaling DbA through the integration of LLMs and graph-theoretic algorithms. The framework employs the Function-Behavior-Structure (FBS) (Gero & Kannengiesser, 2004) ontological framework as its foundational representational paradigm (Goel and Bhatta, 2004, Vandevenne et al., 2016), operating through five distinct computational phases: Retrieval, Mapping, Transfer, Evaluation, and Storage, as illustrated in figure 1.

During the Retrieval phase, LLMs perform systematic extraction of structural and relational information from the design problem specification, encoding it within the FBS ontological framework as a directed dependency graph G(V,E). The vertices V represent functional, behavioral, and structural entities, while edges E encode their interdependencies. Graph-theoretic algorithms, specifically union-find operations, facilitate the identification of functional clusters within G(V,E), enabling the abstraction of these clusters into higher-order functional representations optimized for analogical retrieval.

The Mapping phase implements established analogical reasoning principles (Bhatta and Goel, 1996), wherein LLMs execute cross-domain structural mapping operations through the lens of FBS relationships. This approach ensures preservation of functional isomorphisms despite potential structural

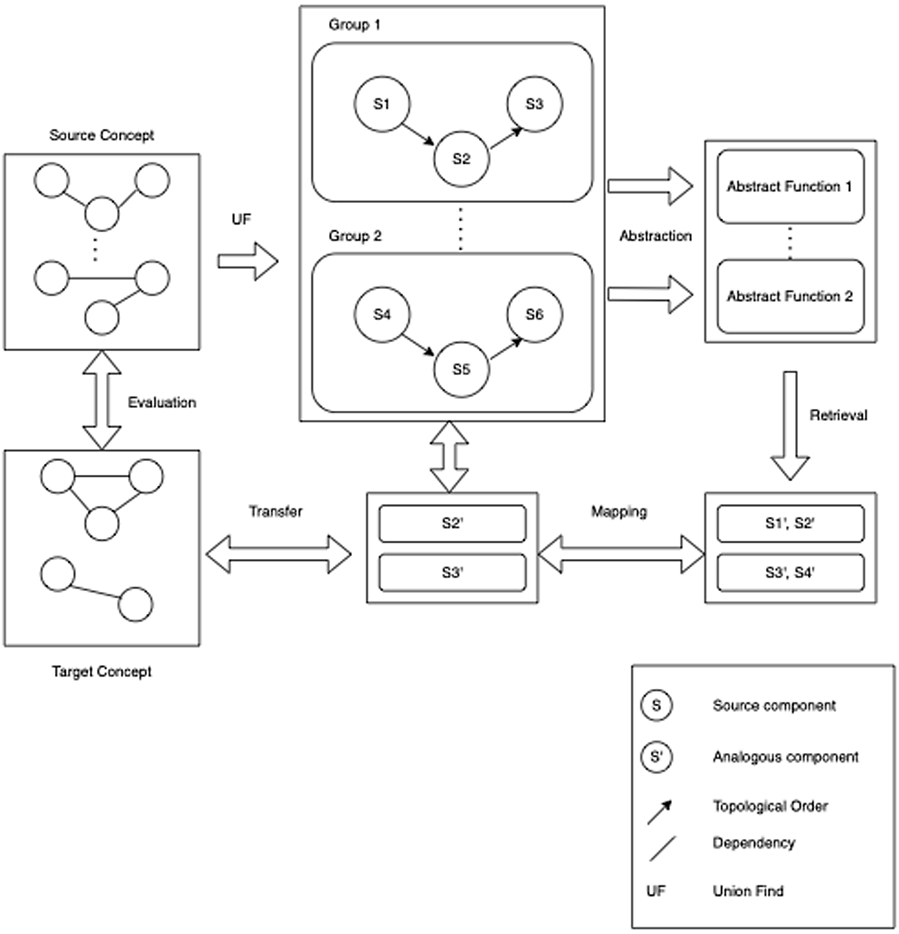

Figure 1. Sequence of operations

variations between domains. The resultant mappings undergo bidirectional projection while maintaining topological consistency with the source design's dependency structure.

The framework employs iterative solution refinement during the Evaluation phase, utilizing quantitative FBS-based metrics to assess functional coherence and problem relevance. The Storage phase culminates in the systematic archival of validated solutions in a normalized FBS representation format, facilitating the development of a comprehensive design analogy repository with robust cross-domain applicability.

3.2. Retrieval phase

The retrieval phase involves leveraging LLMs to extract structural and relational information from the input design problem. This process is divided into two sub-steps:

1. **Identification of Structures and Relationships:** The LLM analyzes the input design context to identify key structural components $S = \{s_1, s_2, \dots, s_n\}$ and the relationships or behaviors $R = \{r_1, r_2, \dots, r_m\}$ between these structures. Each relationship $r_{ij} \in R$ is modeled as a dependency between structures s_i and s_j.
2. **Graph Construction:** The extracted structures and relationships are represented as a directed graph $G = (V, E)$, where:
   - V is the set of vertices, corresponding to the identified structures S,
   - E is the set of directed edges, representing relationships R.

Formally, an edge $e_{ij} \in E$ is defined as:

$$e_{ij} = (s_i, s_j, r_{ij}),$$

where $s_i, s_j \in V$ and r_{ij} encodes the nature of the dependency.

This structured graph G serves as the foundation for downstream reasoning and analogy generation tasks. Figure 2 shows an example graph generated for the components of a motorcycle.

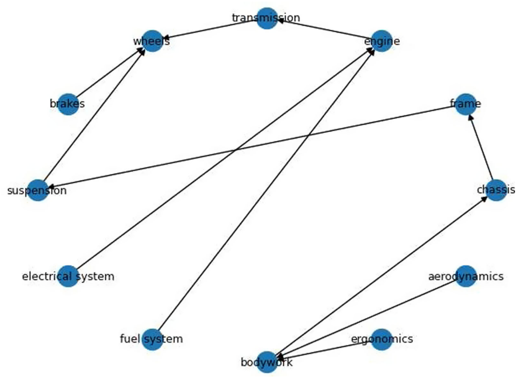

Figure 2. Directed graph depicting the structures and relationships for motorcycle design example

3.3. Component formation using union-find

After constructing the dependency graph $G = (V, E)$, we perform a union-find operation to partition the graph into connected components. Each connected component represents a unique function within the design problem.

3.3.1. Union-Find algorithm

The union-find algorithm is applied to efficiently group nodes into disjoint sets based on their connectivity in G. This process consists of two primary operations:
• **Find:** Determines the representative element (or root) of the set to which a node $v \in V$ belongs. Formally:

$$\mathrm{Find}(v) = \mathrm{Root}(v).$$

• **Union:** Merges two sets containing nodes v_i and v_j if there exists an edge $e_{ij} \in E$ between them. This operation is defined as:

$$\mathrm{Union}\left(v_i, v_j\right) \Rightarrow \mathrm{Root}(v_i) = \mathrm{Root}\left(v_j\right).$$

3.3.2. Component formation

Using the union-find operations, the nodes V are grouped into disjoint sets $C = \{C_1, C_2, \ldots, C_k\}$, where each set C_i corresponds to a connected component. Formally, a connected component C_i is defined as:

$$C_i = \{v \in V | \mathrm{Find}(v) = \mathrm{Root}(v)\}.$$

Each component C_i represents a unique function, encapsulating the structural elements and their relationships that contribute to that specific functionality. This step mirrors methods for function-based clustering Fu et al., 2015.

3.4. Abstract function creation

After grouping the nodes into connected components $C = \{C_1, C_2, \ldots, C_k\}$ using the union-find operation, we assign an abstract function to each component. These abstract functions encapsulate the collective behavior and structural dependencies of the nodes within their respective groups while preserving the original graph topology.

3.4.1. Abstract function representation

Each connected component C_i is mapped to an abstract function F_i, where:

$$F_i = \mathrm{AbstractFunction}(C_i),$$

and C_i represents the set of nodes $\{v_1, v_2, \ldots, v_n\}$ and their internal dependencies. The abstract function F_i is designed to generalize the behavior of the component while hiding low-level implementation details. Figure 3 shows the identified abstract functions for each connected component from figure 2.

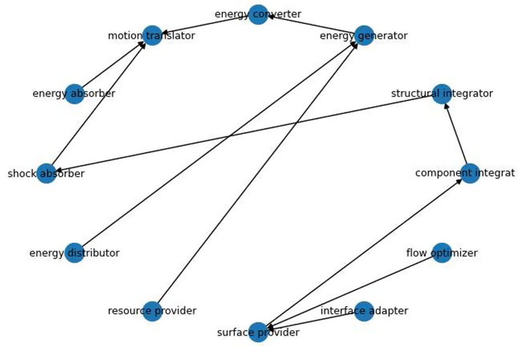

Figure 3. Functions graph preserving the topology of the structure graph - motorcycle design example

3.4.2. Graph transformation

The dependency graph $G = (V, E)$ is transformed into a higher-level abstract graph $G' = (V', E')$, where:
• $V' = \{F_1, F_2, \ldots, F_k\}$ represents the set of abstract functions,
• E' represents the dependencies between abstract functions, derived from the original graph G.
An edge $e'_{ij} \in E'$ is created between F_i and F_j if there exists at least one edge $e_{xy} \in E$, where $v_x \in C_i$ and $v_y \in C_j$. Formally:

$$e'_{ij} = \left(F_i, F_j\right) \quad \text{if} \quad \exists\, e_{xy} \in E,\ v_x \in C_i,\ v_y \in C_j.$$

3.4.3. Preserving topology

The abstract graph G' retains the original graph's topology, ensuring that the hierarchical structure of functions and their dependencies remains consistent. This abstraction facilitates the application of reasoning and design analogy in subsequent phases.

3.5. Retrieval of analogical structures

With the abstract functions $F = \{F_1, F_2, \ldots, F_k\}$ defined for each component, the next step involves leveraging a Large Language Model (LLM) to retrieve analogical structures from various domains that exhibit similar functionality.

3.5.1. Analogical retrieval framework

For each abstract function F_i, the LLM is queried to identify structures $A_i = \{a_1, a_2, \ldots, a_m\}$ from diverse domains that are analogous to F_i in terms of functionality. The retrieval process can be formalized as:

$$A_i = \mathrm{LLM} - \mathrm{Retrieve}(F_i, D),$$

where D represents the set of available domains, and $\mathrm{LLM} - \mathrm{Retrieve}$ is the retrieval mechanism powered by the LLM. The retrieved structures A_i are ranked based on their functional similarity to F_i.

3.5.2. Functional similarity metric

To ensure the retrieved structures are relevant, a functional similarity metric $\mathrm{Sim}\left(F_i, a_j\right)$ is computed for each candidate $a_j \in A_i$. The similarity score is derived from the LLM's embeddings and is defined as:

$$\mathrm{Sim}\left(F_i, a_j\right) = \cos\left(\mathrm{Embed}(F_i), \mathrm{Embed}\left(a_j\right)\right),$$

where $\mathrm{Embed}(\cdot)$ denotes the LLM-generated embedding of the input, and $\cos(\cdot,\cdot)$ is the cosine similarity function. This is reminiscent of techniques such as those used by Chiu and Shu, 2007.

3.5.3. Selection of analogical structures

Based on the similarity scores, the top p structures $A_i^{\mathrm{top}} \subseteq A_i$ are selected for each abstract function F_i:

$$A_i^{\mathrm{top}} = \{a_j \in A_i | \mathrm{Sim}\left(F_i, a_j\right) \geq \tau\},$$

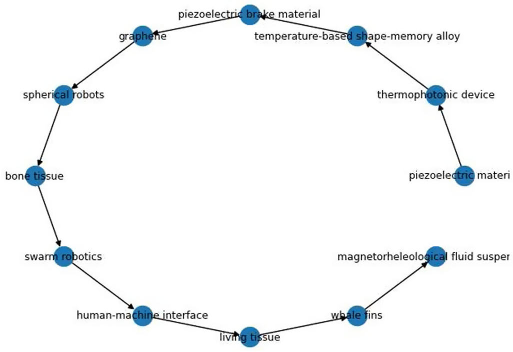

Figure 4. Graph with analogous structures for motorcycle design example

where τ is the similarity threshold. These selected structures represent the most relevant analogies for F_i across domains, similar to PCA-based methods in Verhaegen et al. (2011).

3.5.4. Preservation of abstract graph topology

The retrieved analogical structures $A^{top} = \{A_1^{top}, A_2^{top}, \ldots, A_k^{top}\}$ are integrated back into the abstract graph G' while preserving its topology. Each node F_i in G' is replaced by its corresponding analogical structure A_i^{top}, maintaining the edge connections E' between abstract functions. Figure 4 shows an example for the analogous structures for each abstract function identified in figure 3.

3.6. Mapping and transfer

In the mapping and transfer phase, we proceed in topological order to transfer the function, behavior, and structure from the retrieved analogical structures to the source design problem. The goal is to ensure that the subsequent nodes in the analogy graph remain compatible with each other by respecting the topological dependencies.

3.6.1. Topological ordering

Given the abstract graph $G' = (V', E')$ of the source design problem, we first compute a topological order of the abstract functions $F = \{F_1, F_2, \ldots, F_k\}$. The topological order π is defined as a sequence of functions such that for every directed edge $e'_{ij} = (F_i, F_j) \in E'$, F_i appears before F_j in π. Formally:

$$\pi = \text{TopologicalOrder}(G'),$$

where $\pi = \{F_{\pi(1)}, F_{\pi(2)}, \ldots, F_{\pi(k)}\}$ is a valid topological order.

3.6.2. Function, behavior, and structure mapping

For each function F_i in the topologically ordered sequence π, we map the function F_i, its associated behavior b_i, and its structure s_i to the corresponding components A_i^{top} of the retrieved analogical structure. The mapping process is formally defined as:

$$\text{Map}(F_i) \to \{A_i^{top}, b_i, s\}.$$

This ensures that the function F_i from the source design is transferred to the analogous structure, preserving both its behavior and structure.

3.6.3. Ensuring compatibility

By following the topological order, we ensure that each node F_i is mapped before its dependent nodes F_j (where $F_i \to F_j$ in E'). This guarantees that the behavior and structure of F_i are compatible with those of F_j, ensuring that the subsequent mappings do not violate any functional dependencies within the analogy graph.

3.6.4. Transfer process

The transfer is performed iteratively as follows:

• For each function F_i in the topological order π, retrieve its corresponding analogical structure A_i^{top} from the analogy graph.

• Transfer the structure s_i, behavior b_i, and function F_i from A_i^{top} to the source design problem, ensuring that the analogical structure aligns with the source.
• This process continues until all functions, behaviors, and structures are mapped and transferred to the source.

3.7. Solution storage and future retrieval

The mapping and transfer phase yields a set of design solutions by analogy, each of which encapsulates a functional, behavioral, and structural mapping from the analogical structures to the source design problem. These solutions are then stored in a vector database for efficient future retrieval.

3.7.1. Solution representation

Each solution S_i is represented as a vector in a high-dimensional embedding space. The vector $\mathbf{S_i}$ encodes the combined function, behavior, and structure of the analogical solution:

$$\mathbf{S_i} = \mathrm{Embed}(F_i, b_i, s_i),$$

where F_i is the function, b_i is the behavior, and s_i is the structure of the mapped solution. The embedding function $\mathrm{Embed}(\cdot)$ generates a vector representation for each solution that captures its key characteristics in the design space.

3.7.2. Vector database storage

All solutions $S = \{\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S_n}\}$ are stored in a vector database, such as FAISS or Pinecone, which supports efficient similarity search operations. The vector database allows for fast retrieval of solutions based on their functional similarity to new design problems. Formally, for a new query vector $\mathbf{Q}$ representing a new design problem, the closest solutions $\mathbf{S_i}$ can be retrieved using a similarity metric:

$$\mathbf{S_i}^{\mathrm{best}} = \mathrm{Retrieve}(\mathbf{Q}, S),$$

where $\mathbf{S_i}^{\mathrm{best}}$ represents the solution most similar to the query vector $\mathbf{Q}$, and $\mathrm{Retrieve}(\cdot)$ is the retrieval function based on cosine similarity or another distance metric.

3.7.3. Future retrieval

By storing the solutions in a vector database, future design problems can be efficiently matched with the most relevant analogical solutions. This retrieval process enables quick adaptation of previous analogical designs to new contexts, facilitating faster design iterations and improving design efficiency over time.

4. Results

4.1. Case study - motorcycle design

The result obtained from the proposed framework for 'Motorcycle Design' is presented in Table 1 (edited to tabular format for clarity)
The results of the proposed framework present a structured, systematic, and modular approach to motorcycle design. By breaking down the vehicle into distinct subsystems and then solving by analogy, it ensures that every aspect of the motorcycle's functionality—energy generation, conversion, translator, shock absorber, etc.—is optimized individually. It integrates advanced engineering concepts such as piezoelectric energy harvesting, thermophotonic devices, shape-memory alloys, and graphene-based systems. This method ensures a high level of technical precision and modularity, making it well-suited for real-world engineering applications where individual subsystems can be optimized independently.

Table 1. Motorcycle System Overview

Step	Description	Function	Behavior	Structure
Step 1: Resource Provision	The motorcycle is equipped with a bio-inspired, **piezoelectric material-based system** integrated into the frame, harnessing vibrational energy from the engine and converting it into electrical energy. This system supplements the primary power source, increasing overall efficiency.	**Resource provision**	Energy harvesting and conversion	Piezoelectric material-based system
Step 2: Energy Generation	The motorcycle utilizes a novel, **thermophotonic device** that converts heat into energy, harnessing waste heat and converting it into electrical energy. This device supplements the primary power source, increasing overall efficiency.	**Energy generation**	Heat-to-energy conversion	Thermophotonic device
Step 3: Energy Conversion	The motorcycle employs a **shape-memory alloy-based system** that adjusts gear ratios in response to changes in temperature, providing seamless and efficient gear shifting.	**Energy conversion**	Temperature-responsive gear shifting	Shape-memory alloy-based system
Step 4: Energy Absorption	The motorcycle is equipped with a regenerative braking system inspired by piezoelectric materials, **harnessing kinetic energy** generated during braking and **converting it into electrical energy**.	**Energy absorption**	Kinetic energy harvesting and conversion	Regenerative braking system
Step 5: Energy Distribution	The motorcycle utilizes a **graphene-based power distribution system** that optimizes energy flow and reduces energy loss, providing efficient and reliable power distribution.	**Energy distribution**	Optimized energy flow	Graphene-based power distribution system
Step 6: Motion Translation	The motorcycle employs a **novel, omnidirectional wheel system** inspired by **spherical robots**, providing unparalleled maneuverability and stability.	**Motion translation**	Omnidirectional movement	Omnidirectional wheel system
Step 7: Structural Integration	The motorcycle is designed with a **3D-printed, lattice-structured frame** inspired by **bone tissue**, providing exceptional strength, lightness, and flexibility.	**Structural integration**	Load-bearing and flexibility	3D-printed, lattice-structured frame
Step 8: Component Integration	The motorcycle utilizes a **modular, adaptive chassis system** inspired by **swarm robotics**, providing exceptional flexibility and adaptability.	**Component integration**	Modular and adaptive configuration	Modular, adaptive chassis system
Step 9: Interface Adaptation	The motorcycle is designed with a **biometric sensor-based interface** inspired by human-machine interfaces, providing real-time feedback and adapting to the rider's physical and emotional state.	**Interface adaptation**	Real-time feedback and adaptation	Biometric sensor-based interface
Step 10: Surface Provision	The motorcycle is designed with a **self-healing, adaptive skin** inspired by **living tissues**, providing exceptional durability, aerodynamics, and aesthetics.	**Surface provision**	Self-healing and adaptation	Self-healing, adaptive skin
Step 11: Flow Optimization	The motorcycle is designed with a **bio-inspired, adaptive aerodynamic system** inspired by **whale fins**, providing exceptional aerodynamic efficiency.	**Flow optimization**	Aerodynamic efficiency	Bio-inspired, adaptive aerodynamic system
Step 12: Shock Absorption	The motorcycle utilizes a **magnetorheological fluid-based suspension system** inspired by shape-memory alloys, providing exceptional shock absorption, stability, and adaptability.	**Shock absorption**	Adaptive shock absorption	Magnetorheological fluid-based suspension system

5. Discussion

This work presents a theoretical framework integrating LLMs and graph algorithms within the FBS ontology for DbA. The framework systematically addresses analogical mapping while maintaining functional consistency through structured graph representations. The utilization of FBS ontology as a common semantic framework enables cross-domain translation, addressing a fundamental challenge in analogical reasoning.

The incorporation of LLMs represents a methodological shift from conventional approaches dependent on curated databases or domain-specific knowledge bases. This enables exploration of a broader solution space without extensive data collection requirements. However, this approach necessitates careful consideration of prompt engineering methodologies, potential memorization artifacts in LLM outputs, and development of quantitative metrics for evaluating analogical relevance.

The implementation of graph algorithms, specifically union-find operations and topological ordering, provides a formal mechanism for preserving structural consistency in compound analogical structures. While this approach establishes mathematical rigor in maintaining functional relationships, further investigation is required regarding optimal graph representations for diverse design problems and computational scalability for complex dependency structures.

Several theoretical and methodological considerations emerge from this conceptual framework:

1. **Computational complexity:** The scalability of graph operations and LLM query optimization requires systematic evaluation, particularly for extensive dependency networks.
2. **Ontological framework:** The efficacy of FBS as a universal translation mechanism across heterogeneous domains demands rigorous investigation.
3. **Current development:** Ongoing research focuses on optimizing LLM query formulation and graph transformation operations, including the development of quantitative metrics for analogical relevance and implementation of vector-based solution storage systems.

As a conceptual framework, this research contributes to the DbA domain through a theoretical foundation for automated analogical reasoning. The synthesis of FBS ontology, LLMs, and graph algorithms establishes a systematic methodology for cross-domain analogical mapping while maintaining functional consistency. However, substantial research remains in empirical validation, metric development, and addressing technical constraints in LLM reliability and graph representations.

Future research directions include systematic framework evaluation and theoretical refinement. Additionally, investigation of framework behavior across diverse design domains will provide insights into generalizability constraints.

6. Conclusion

This paper presents a theoretical framework for DbA that combines LLMs with graph algorithms. The framework employs the FBS ontology as a basis for cross-domain translation, supported by a mathematical formulation for maintaining structural dependencies. By representing design problems as dependency graphs and utilizing union-find operations for functional clustering, the framework provides a systematic approach to handling compound analogical structures.

The integration of LLMs with graph-theoretic operations offers a mechanism for exploring cross-domain analogies while preserving functional relationships. The framework's formalization of the retrieval, mapping, and transfer processes establishes a theoretical foundation for systematically generating and evaluating design analogies.

While the framework demonstrates potential in automating DbA, certain limitations should be acknowledged. The quality of analogical retrieval depends significantly on the LLM's training and its ability to understand domain-specific technical concepts. Additionally, the framework's current formulation assumes that functional relationships can be effectively captured through graph structures, which may not hold true for all design scenarios.

Future research directions include the development of robust evaluation metrics for assessing the quality of retrieved analogies, investigation of methods to incorporate domain-specific constraints into the mapping process, and exploration of techniques to handle temporal and dynamic aspects of design problems. The framework could also benefit from integration with existing design tools and methodologies to enhance its practical applicability in real-world design scenarios.

References

Aamodt, A., & Plaza, E. (1994). Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Communications*, 7(1), 39–59.

Ball, L. J., & Christensen, B. T. (2022). Analogical reasoning and mental simulation in design: Two strategies linked to uncertainty resolution. In *About designing* (pp. 137–152). CRC Press.

Bhatta, S. R., & Goel, A. K. (1996). From design experiences to generic mechanisms: Model-based learning in analogical design. *AI EDAM*, 10(2), 131–136.

Briana, L., Julie, L., Turner, C., *et al.* (2015). Design repository&analogy computation via unit-language analysis (dracula) matching algorithm development.

Chiu, I., & Shu, L. (2007). Biomimetic design through natural language analysis to facilitate crossdomain information retrieval. *Ai Edam*, 21(1), 45–59.

Fu, K., Murphy, J., Yang, M., Otto, K., Jensen, D., & Wood, K. (2015). Design-by-analogy: Experimental evaluation of a functional analogy search methodology for concept generation improvement. *Research in Engineering Design*, 26, 77–95.

Gero, J. S., & Kannengiesser, U. (2004). The situated function–behaviour–structure framework. *Design studies*, 25(4), 373–391.

Goel, A. K., & Bhatta, S. R. (2004). Use of design patterns in analogy-based design. *Advanced Engineering Informatics*, 18(2), 85–94.

Goel, A. K., de Silva Garza, A. G., Grue, N., Murdock, J. W., & Recker, M. M. (1997). Functional explanations in design. W980420122, 221.

Grace, K., Gero, J., & Saunders, R. (2015). Interpretation-driven mapping: A framework for conducting search and rerepresentation in parallel for computational analogy in design. *AI EDAM*, 29(2), 185–201.

Holyoak, K. J., & Thagard, P. (1996). *Mental leaps: Analogy in creative thought*. MIT press.

Hybs, I., & Gero, J. S. (1992). An evolutionary process model of design. *Design Studies*, 13(3), 273–290.

Mubarak, K. (2004). Case based reasoning for design composition in architecture. *Pittsburgh, Carnegie Melon University.*

Oriakhi, E., Linsey, J., & Peng, X. (2011). Design-by-analogy using the wordtree method and an automated wordtree generating tool. *DS 68-7: Proceedings of the 18th International Conference on Engineering Design (ICED 11), Impacting Society through Engineering Design, Vol. 7: Human Behaviour in Design, Lyngby/ Copenhagen, Denmark, 15.-19.08. 2011.*

Perner, P. (2014). Mining sparse and big data by case-based reasoning. *Procedia Computer Science*, 35, 19–33.

Qin, X., & Regli, W. C. (2003). A study in applying case-based reasoning to engineering design: Mechanical bearing design. *AI EDAM*, 17(3), 235–252.

Sanaei, R., Lu,W., Blessing, L. T., Otto, K. N., &Wood, K. L. (2017). Analogy retrieval through textual inference. *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 58127, V02AT03A007.

Singh, V., Casakin, H., *et al.* (2015). Developing a computational framework to study the effects of use of analogy in design on team cohesion and team collaboration. *DS 80-11 Proceedings of the 20th International Conference on Engineering Design (ICED 15) Vol 11: Human Behaviour in Design, Design Education; Milan, Italy, 27-30.07. 15*, 101–110.

Song, H., Evans, J., & Fu, K. (2020). An exploration-based approach to computationally supported design-by-analogy using d3. *AI EDAM*, 34(4), 444–457.

Stone, R. B., & Wood, K. L. (1999). Development of a functional basis for design. *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 19739, 261–275.

Vandevenne, D., Verhaegen, P.-A., Dewulf, S., & Duflou, J. R. (2016). Seabird: Scalable search for systematic biologically inspired design. *Ai Edam*, 30(1), 78–95.

Vattam, S., Wiltgen, B., Helms, M., Goel, A. K., & Yen, J. (2011). Dane: Fostering creativity in and through biologically inspired design. *Design creativity 2010*, 115–122.

Verhaegen, P.-A., Peeters, J., Vandevenne, D., Dewulf, S., & Duflou, J. R. (2011). Effectiveness of the panda ideation tool. *Procedia engineering*, 9, 63–76.