Computablizing Human Knowledge for Machines through Domain-Specific Representation

Yu-Zhe Shi, Zhangqian Bi, Lecheng Ruan, Yao Wan, Huamin Qu,
Qining Wang, *Senior Member, IEEE*, Philip S. Yu, *Life Fellow, IEEE*

***Abstract*—Across science, engineering, medicine, governance, and other expertise-intensive domains, practitioners increasingly adapt general-purpose Artificial Intelligence (AI) to specialized tasks by encoding domain knowledge in explicit, machine-processable artifacts, including data schemas, notation systems, formal grammars, declarative models, and executable specifications. In this survey, we use Domain-Specific Representations (DSRs) as a functional umbrella term for these established artifact families, grouping them by their role in making domain knowledge computable rather than by a shared syntax or formalism. Since such artifacts are often developed for individual applications, domain context and methods for incorporating knowledge into AI become intertwined, resulting in fragmented, case-specific solutions with few transferable practices. We present a systematic cross-domain analysis of AI approaches that explicitly incorporate DSRs, organized along two dimensions: (i) the function of the encoded knowledge, classified from a Bayesian perspective as generative, discriminative, or descriptive; and (ii) the mechanism by which AI uses that knowledge—retrieval, modularization, composition, decomposition, planning, optimization, or explanation. The framework maps the literature, identifies recurring challenges and research opportunities, and reveals shared principles for designing, deploying, and communicating DSRs, thereby supporting more systematic development and broader adoption of AI across expertise-intensive domains.**

***Index Terms*—Information Sciences, Human Knowledge Computablization, Domain-Specific Representation.**

I. Introduction

Four centuries ago, Johannes Kepler dedicated years of painstaking manual calculation to derive his laws of planetary motion from Tycho Brahe's voluminous observational records [1]. Today, analogous pattern discovery in astronomical data can be achieved within minutes or even seconds, facilitated by modern computational methods [2], notably Artificial Intelligence (AI), with breakthroughs in Artificial Neural Network (ANN) architectures [3], optimization techniques for data-driven training [4], methods for collecting and preprocessing large-scale training datasets [5], and novel hardware for efficient parallel computing and memory management [6], [7]. Inspired by these transformative advances in AI, practitioners in critical sectors including scientific discovery [8], engineering innovation [9], social governance [10], medical practice [11], information governance [12], and computational arts [13] have been actively integrating AI approaches into their conventional workflows to advance the frontiers of their respective domains (see Fig. 1A).

Despite the general (*i.e.*, application-agnostic) capabilities of AI solutions, an intrinsic gap persists between the outputs of AI research and their practical application to specific domain contexts. The former adheres to a *method-centric* methodology, focusing on developing and evaluating algorithms where task objectives and input-output relationships are abstracted from a series of real-world problems that differ superficially (*e.g.*, in problem context) but share underlying commonalities (*e.g.*, abstract formulation). This approach seeks to achieve the broadest possible scope, reflecting the fundamental methodology of computer science [14]. The latter, by contrast, follows a *requirement-centric* methodology that emphasizes the contextual details sacrificed in such abstraction [15]. These details include the structure of the input-output space, which specifically reflects the properties and principles of the domain and proves critical in domain-specific applications. Let us consider Kepler's case: when contemporary astronomers attempt to induce physics expressions from orbital records, they may leverage state-of-the-art regression models trained extensively on sequential data, but they cannot employ these models in a plug-and-play manner [2]. Instead, they have to manually craft a syntax of physics symbols and operators, layering this syntax atop the original models to constrain the search space and ensure that the resulting expressions remain valid according to domain-specific astronomical knowledge. Without such syntax, the search space over the combinations of characters would be overwhelmingly large, and most resulting expressions would prove meaningless.

This approach of crafting artifacts for specific domain applications using general AI solutions has gradually become a common practice due to various considerations. For instance, prior knowledge of the mechanisms by which viruses derive mutations is exploited to simulate viral escape from the human immune system [16]. In this example, the prior grammar describes rules derived from domain-specific immunological knowledge and is used to generate a space of meaningful candidate results within the general embedding vector space provided by AI models. In motion graphics rendering, a set of logic programs is employed to validate that joint articulation is properly generated subject to motion dynamics in the physical world [17]. In this application, the logic predicates capture physical knowledge of motion dynamics to discriminate whether AI-generated trajectories are physically plausible.

This work is partially supported by the National Natural Science Foundation of China under Grants 625B2160 and 52475001, and RGC GRF Grant 16210722. Yu-Zhe Shi and Zhangqian Bi contributed equally. (Corresponding authors: Lecheng Ruan, Yao Wan, and Qining Wang)

Yu-Zhe Shi and Huamin Qu are with Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong.

Zhangqian Bi and Yao Wan are with School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China.

Lecheng Ruan, and Qining Wang are with School of Advanced Manufacturing and Robotics, Peking University, Beijing, China.

Philip S. Yu is with Department of Computer Science, University of Illinois Chicago, Chicago, IL.

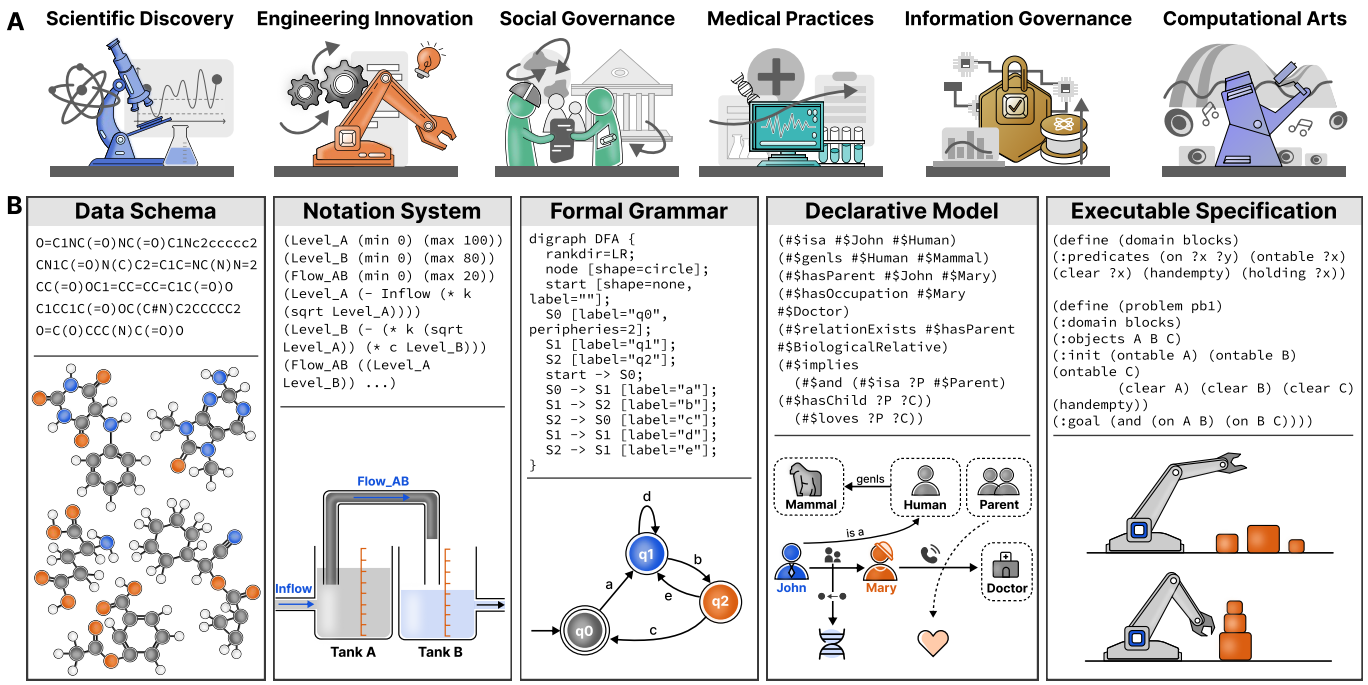


Fig. 1. **Overview of DSRs and their applications. (A)** Visual illustrations of the six expertise-intensive critical sectors. **(B)** Demonstrations of the five implementations of DSRs and their representative applications, with exemplar code snippets **(top)** and the physical instances they mirror **(bottom)**.

Another example exploits a compiler for a chemical programming language that restricts experimental procedures translated from Natural Language (NL)-based documents to machine-executable formats in a self-driving laboratory [18]. In this case, the programming language integrates the engineering ontology of device hardware properties, thereby establishing the plausible output space for AI document translation. Although in these cases the domain-expert-crafted artifacts play different roles in interacting with AI, they all enable AI to exploit fine-grained, self-contained, and verifiable human knowledge that is difficult to incorporate during the development of these AI approaches themselves, thereby bridging the gap for applying these AI approaches to specific domain contexts.

The common operation in these examples is to render domain knowledge *computable* for deliberate use in AI workflows. In this survey, we use Domain-Specific Representation (DSR) as an umbrella term for explicit, machine-processable artifacts that encode the intrinsic logic, ontological structures, and computational mechanisms of specific domains [19]. The term groups established artifact families by their shared function in AI workflows rather than by a common representation formalism. These representations encompass several distinct categories: data schemas (*e.g.*, information exchange protocols such as SMILES for molecular computation [20]), notation systems (*e.g.*, mathematical and physical symbology and expressions such as QSIM for dynamic system simulation [21]), formal grammars (*e.g.*, Context-Free Grammars (CFGs) [22] such as Graphviz for graphics composition [23]), declarative models (*e.g.*, logic programs and ontologies such as the CYC knowledge base for commonsense knowledge representation [24]), and executable specifications (*e.g.*, Domain-Specific Languages (DSLs) [25] such as PDDL for robot Task-and-Motion Planning (TAMP) specification [26]). While many broadly applied DSRs, including these representative examples, were not deliberately designed for AI compatibility, they nonetheless facilitate the computational processing of diverse forms of human knowledge within computational frameworks, including AI solutions. These representations support various objectives across different disciplinary contexts, thereby providing domain practitioners with additional options other than manual efforts (see Fig. 1B).

For brevity, we refer to AI approaches that incorporate DSRs as DSR-AI approaches. Although increasingly prevalent, the literature remains largely application-specific, with limited synthesis into transferable engineering principles or broader scientific insights [27]. This fragmentation likely reflects the lack of cross-domain analysis of the AI application ecosystem. Domain experts developing DSR-AI solutions operate from the requirements perspective, focusing solely on applying state-of-the-art AI models to their specific applications. Consequently, in their DSR-AI solutions, the specific domain contexts (*i.e.*, considerations that align with the conventions, requirements, or regulations of the application domain) and the methodologies for incorporating such knowledge with AI (*i.e.*, design choices for solving tasks through the integration of domain knowledge and AI) become intertwined. This status quo preserves the case-specific, labor-intensive nature of DSR-AI approach development, *i.e.*, the expertise and effort invested by the joint cohort of domain experts and information science specialists are difficult to transfer to other applications, thereby hindering the democratization of AI across a broader spectrum of domains, particularly those requiring intelligent computing capabilities where practitioners lack extensive AI backgrounds.

To bridge this gap, this survey develops a cross-domain

framework that separates *what* domain knowledge expresses from *how* AI uses it, rather than proposing a new knowledge representation methodology or programmable artifact type. Specifically, we first review the history of computablizing human knowledge for AI and depict the landscape of relevant literature (Sec. II). We then present a two-dimensional framework that explicitly spans knowledge function and utilization mechanism, along with its corresponding taxonomy, to establish the foundation of this survey (Sec. III). Subsequently, we conduct an analysis of current DSR-AI approaches using the proposed framework across diverse application domains (Sec. IV). Finally, we highlight the challenges and research opportunities regarding the design, deployment, and communication of DSRs for AI (Sec. V), and conclude this survey with a general discussion of relevant arguments and defenses (Sec. VI). This survey is expected to generate increased attention, induce constructive principles for practice, stimulate fruitful discussions, and inspire further research ideas.

II. Preliminaries

This section begins with the inception of thoughts and practices on computablizing human knowledge to make AI approaches applicable to specific domains (Sec. II-A), along with the historical development of the artifact families grouped as DSRs (Sec. II-B) and DSR-AI methodologies (Sec. II-C). Subsequently, we review related surveys that address the high-level concept of knowledge computablization and discuss the significance of this survey (Sec. II-D). Finally, we report the protocol for paper selection and the resulting bibliometric details of the selected literature corpora (Sec. II-E).

A. AI and the Computablization of Human Knowledge

The aspiration to render human knowledge computationally tractable for AI dates back to the earliest foundations of AI research. In the 1950s–1970s, symbolic AI pioneers such as Newell and Simon demonstrated that logical reasoning could be formalized through computational structures, exemplified by the *Logic Theorist*, which was capable of proving mathematical theorems [28]. This established the core paradigm of knowledge representation: encoding human expertise into machine-interpretable formats to enable automated reasoning. By the 1980s, expert systems (*e.g.*, MYCIN for medical diagnosis [29]) operationalized this approach by embedding domain-specific rules (*e.g.*, *"IF bacterial infection THEN prescribe antibiotics"*) into inference engines. However, these early systems faced significant limitations: handcrafted rules proved brittle for complex real-world scenarios, and the "knowledge acquisition bottleneck" hindered the scalability of AI solutions [30]. Specifically, as the complexity of the target application domain increased exponentially, for example, from medical school quiz-level problems to practices in a comprehensive clinic, the complexity of expert systems could not scale linearly due to the limited modularity of domain-specific rules, resulting in "big balls of mud" [31].

The rise of statistical machine learning (1990s–2000s), featuring representative models such as ANNs [32], Support Vector Machines (SVMs) [33], and Decision Trees (DTs) [34], shifted focus toward data-driven pattern recognition. This ushered in an era in which decision features no longer needed to be fully hand-engineered but could instead be *learned* to fit a given data distribution. During this period, the integration of domain knowledge remained critical for guiding models to align with domain consensus and for augmenting their input-output space. Bayesian networks, for instance, formalized causal relationships using probability distributions grounded in domain expertise [35]. Similarly, semantic web technologies (*e.g.*, OWL ontologies [36]) enabled structured knowledge sharing across domains. Additionally, in biochemistry, COBRA embedded reaction stoichiometry as linear constraints to simulate metabolic fluxes [37]. Hybrid systems such as Markov logic networks bridged first-order logic with statistical learning for uncertain reasoning [38]. This paradigm shift advanced AI solutions beyond brittle handcrafted rules toward more flexible forms of knowledge representation that capable of adapting to the uncertainty introduced by data fitting and probabilistic inference. However, AI models still required fine-grained manual design, with knowledge integration and model construction remaining highly intertwined, making AI applications largely "a game for the few".

The rise of Deep Neural Networks (DNNs) (2010s–present) intensified implicit knowledge extraction from large-scale datasets but amplified the need for explicit guidance and guardrails. For DNNs [39], models with strong feature extraction capabilities can be misguided by data patterns without sufficient domain knowledge, even when facing slightly distorted data distributions caused by noisy or limited samples (*e.g.*, a trained cancer imaging classifier may learn a shortcut based on labels created at the data source, such as annotations made by staff at the original hospital [40]). In response, physics-informed neural networks embedded differential equations as regularization terms within the overall optimization objective, solving fluid dynamics problems with 100 times less data [41]. More advanced DNN models, such as Transformer [3], and the resulting scaled models, Large Language Models (LLMs) [4], take this approach to the extreme with large-scale data from all possible source domains, tasks, and data modalities. These models are thought to capture all kinds of human knowledge and can be generally accepted as a commonsense knowledge base or a "world model" [42]. However, although LLMs capture a broad range of human knowledge, they may not efficiently elicit such knowledge due to their non-deterministic nature, which can lead to hallucination in output [43].

B. Historical Development of DSR Artifact Families

The artifact families grouped as DSRs in this survey developed from a common need: to transform rich, often implicit, and context-bound knowledge in specialized domains into forms suitable for computational processing. This necessity arose from the significant limitations encountered when applying generic data structures and general-purpose programming languages to complex fields such as engineering design or medical diagnosis. Consequently, DSRs evolved as purpose-built artifacts designed to explicitly encode the essential *knowledge fabric* of a domain, including its core entities and their

interrelations, governing principles and rules, and allowable transformations and processes. Practitioners facing concrete challenges in knowledge sharing, automating task completion, or enabling logical reasoning developed specialized representational methods [19]. These methods matured into five families of DSRs—data schemas, notation systems, formal grammars, declarative models, and executable specifications—each providing distinct methodological approaches to make implicit domain knowledge explicit, structured, and ultimately operable by machines.

Data schemas represent one of the earliest and most practical forms of DSRs, originating from the fundamental need for structured data exchange within and between organizations. Their development is closely tied to database management and electronic data interchange. Early implementations such as hierarchical and network database schemas evolved into the relational model formalized by Codd [44], settling the foundation of structured querying. The rise of the internet and complex systems required more flexible and expressive schemas, leading to XML Schema Definition for document validation and data exchange protocols [45], and later JSON Schema for web Application Programming Interfaces (APIs) [46]. The core methodology involves defining types, constraints, relationships, and validation rules for structured data instances. The advantages of data schemas include interoperability, enforcement of data integrity, and clear structure understandable by both humans and machines. Their disadvantages include high rigidity (particularly with XML Schema), potential complexity that can hinder adoption, and difficulties in representing complex constraints or dynamic knowledge beyond static structures.

Notation systems constitute a fundamental DSR family developed for the computational use of knowledge, particularly in mathematical and physical systems. These systems encode domain concepts and relationships through specialized symbols and syntactic rules, such as mathematical notation, chemical formulas, Feynman diagrams, and musical notation, *etc*. The methodology focuses on defining a precise, unambiguous symbolic language grounded in the domain's semantics. Notation systems has been formalized through computer algebra systems that enable symbolic computation and theorem proving [47]. Notable examples include Schoonschip, which was primarily used by the European Organization for Nuclear Research [48]; Maple, which is predominantly employed in non-linear systems such as computing symbolic solutions to Partial Differential Equations (PDEs) [49]; and Mathematica, which is widely utilized in engineering computation [50]. The advantages of notation systems include conciseness, expressive power for experts, and deep integration with domain theory. The disadvantages include inherent ambiguity in some notations (requiring contextual interpretation), steep learning curves, and challenges in fully automating reasoning due to the need for deep semantic understanding beyond mere syntax.

Formal grammars represent a distinct and foundational family of DSRs primarily concerned with the precise specification of syntactic structure within a domain's symbolic or textual expressions. Rooted in formal language theory, particularly the Chomsky hierarchy, and compiler technology, their role as DSRs centers on defining the rules that govern how valid sequences of symbols or tokens can be combined to form well-structured expressions specific to a domain. Methodologically, formal grammars, especially CFGs and their extensions such as Attribute Grammars (AttrGs) or Parsing Expression Grammars (PEGs), employ production rules. These rules systematically describe how basic terminal symbols (the atomic elements of the language generated by the grammar) can be assembled using non-terminal symbols (representing syntactic categories) into allowable hierarchical structures. The core computational contribution enabled by this DSR family is the rigorous definition of syntax, which facilitates parsing (analyzing input strings to determine grammatical conformity and derive parse trees) and syntactic validation [51]. Tools such as YACC, the first parser generator for UNIX [52]; GNU Bison, the parser generator used in Bash Shell and GCC [53]; and ANTLR, a more recent parser generator employed in contemporary research and development [54], automate the generation of parsers from grammatical specifications, thereby significantly simplifying the implementation of syntax-driven processing tools [55]. However, substantial limitations exist. Pure CFGs inherently struggle to express context-sensitive constraints, which are common in many real-world domains, often requiring complex extensions such as semantic predicates that blur the syntactic-semantic boundary. Designing concise, unambiguous, and maintainable grammars for large, complex domains remains challenging. Importantly, formal grammars primarily address the form of domain expressions and their correct structure, while deferring the critical aspect of meaning (semantics) to subsequent phases of interpretation, compilation, or linkage with other DSRs such as ontologies. Although they often serve as the syntactic foundation for implementing other DSRs, including notation systems or DSLs, formal grammars constitute a distinct class focused exclusively on structural definition.

Declarative models represent a paradigm shift towards specifying what is true or desired in a domain, rather than how to achieve it procedurally. This family includes Logic Programming (*e.g*., Prolog), ontologies (*e.g*., OWL), rule-based systems, and certain knowledge representation frameworks. Their inception was driven by research in automated reasoning and knowledge-based systems. Prolog, developed in the 1970s, allowed encoding domain knowledge as facts and rules, enabling logical inference [56]. Ontologies, gaining prominence with the Semantic Web initiative, use formal logics to define domain concepts, relationships, and individuals in a machine-interpretable way [36]. The methodology emphasizes defining entities, their properties, interrelationships, and constraints declaratively, thereby settles the foundation of the general representation—Knowledge Graphs (KGs) [57]. The advantages of declarative models include the separation of knowledge from computation, support for powerful inference engines to derive implicit knowledge, reusability, and a focus on meaning (semantics). The disadvantages involve the computational complexity of reasoning, the "knowledge acquisition bottleneck" in building comprehensive models, potential brittleness outside modeled constraints, and the difficulty of effectively representing procedural knowledge.

Executable specifications, most notably DSLs, represent DSRs designed not only for description but for direct execution or transformation into executable code within their domain. Their development stems from the desire to raise the level of abstraction for domain experts, allowing them to express solutions or models using familiar domain concepts while being protected from the complexities of general-purpose programming [25]. Early examples include report generators and simulation languages [58]. Modern DSLs encompass a spectrum ranging from internal (embedded) languages or libraries within General-Purpose Languages (GPLs) to external languages equipped with custom toolchains. Representative examples include SQL for database queries [59]; PDDL for robot TAMP [26]; LaTeX [51], IBM SCRIPT Language [60], and Founder Bookmaker Language [61] for document layout at individual and publisher levels; Blender and FreeCAD for modeling 3D structures, objects, and scenes [62], [63]; PyTorch for ANN configuration and training [64]; MATLAB Simulink and Modelica for complex system simulation [65], [66]; and Hardware Description Language (HDL) for digital circuit design [67]. The methodology involves designing languages with syntax and semantics closely aligned to the domain's concepts and workflows, often using formal grammars (for syntax) and declarative or imperative semantics. The advantages of DSLs include significantly improved productivity for domain experts, enhanced verifiability since code is closer to specification, reduced errors by abstracting irrelevant details, and improved communication within the domain. The disadvantages include the cost of designing and maintaining the DSLs and their tooling, potential performance overhead, limited applicability outside the target domain, and the learning curve for new DSLs [68].

C. The Trend of DSR for AI Applications

The integration of human knowledge into AI reflects a fundamental evolutionary trajectory in addressing AI's practical limitations. Initially, practitioners directly encoded human knowledge into rule-based systems such as MYCIN for medical diagnostics, but these systems proved brittle when confronted with real-world complexity and uncertainty. The subsequent shift toward data-driven learning promised greater adaptability, yet it merely transferred the burden: statistical methods like Bayesian networks required laborious pre-design of probabilistic structures, while early machine learning approaches demanded meticulous feature engineering to align models with domain logic. As deep learning pushed data-centric approaches to their limits, ingesting massive datasets to implicitly capture human knowledge, new challenges emerged. Models became unwieldy black boxes, incapable of reliably eliciting or controlling domain-specific knowledge for targeted applications, often generating inconsistent or ungrounded outputs when confronted with specialized tasks.

This trajectory spurred AI researchers to pioneer structured knowledge integration through proof-of-concept studies on classical AI problems. In handwritten digit recognition, Dai *et al*. formalized rules of multi-bit addition and subtraction through Inductive Logic Programming (ILP), reducing error rates by over 45% compared to state-of-the-art DNN-based classifiers [69]. For object relation recognition, Mao *et al*. introduced compositional syntax that decomposed objects into semantic components, such as shape and color, enhancing interpretability while maintaining approximately 40% higher accuracy than benchmark records [70]. In pictogram composition, Feinman and Lake integrated stroke and part-location priors to balance the randomness and plausibility of drawing generation, improving likelihood by approximately two-fold over prior-free counterparts [71]. These innovations demonstrated that even for relatively "toy-scale", application-agnostic AI tasks, frameworks that integrate AI with explicit human knowledge representation, namely DSRs, could enhance robustness, efficiency, and explainability. This undoubtedly provides encouraging evidence for domain-specific applications, serving as methodological validation through general, abstract problems. This validation confirms that the fragmented attempts at DSR-AI approaches for specific applications that emerged around the start of the 2020s represent more than isolated "aha! moments". Instead, they signal the dawn of a new era in making AI accessible to all human practitioners.

Inspired by these foundations, domain practitioners catalyzed a DSR-AI boom from 2018 onward, with publications incorporating domain-specific constraints surging by over 10 times across diverse practical application domains. This acceleration intensified with the advent of LLMs: while their broad knowledge scope impressed experts, their limitations in adapting to and executing specialized workflows revealed critical gaps, highlighting DSR's potential contributions to AI, which include—but are not limited to—(i) constraining generated results to be meaningful within the domain and obtaining such results efficiently without exploring excessively large search spaces; (ii) maintaining consistency and completeness of results to achieve deterministic or explainable non-deterministic outcomes; and (iii) facilitating consideration of fine-grained domain details through targeted domain knowledge injection.

D. Related Surveys

The concept of integrating formalized human knowledge with AI approaches has been explored from various perspectives in prior literature. We identify related surveys through three established viewpoints: (i) the theoretical perspective, which summarizes theories for incorporating prior knowledge into Machine Learning (ML) algorithms; (ii) the technical perspective, focusing on Neuro-Symbolic Artificial Intelligence (NeSyAI), a mainstream family of approaches that integrates knowledge with ANNs for various AI tasks; and (iii) the application perspective, which addresses artifacts that enhance domain knowledge for AI solutions in specific domains. We discuss both the contributions and limitations of these surveys, thereby highlighting the unique significance of our work.

Several surveys have investigated the integration of prior domain knowledge into ML and Deep Learning (DL), also known as "informed machine learning". Von Rueden *et al*. introduce a taxonomy for informed machine learning, categorizing theories by the source and representation of knowledge and the stage at which it is integrated into the ML pipeline [72].

They emphasize that knowledge can take many forms (*e.g*., logic rules, simulation data, semantic graphs) and can enter the learning process as additional features, constraints, or priors. Dash *et al*. focus specifically on strategies for injecting domain knowledge into DNNs, demonstrating that simple forms of knowledge (such as logical constraints or invariants) incorporated into DNNs can substantially improve model performance and reliability [73]. Building on this foundation, Cui *et al*. further define a framework to identify relevant domain knowledge, represent it formally, and integrate it into DNNs [74]. Unlike earlier works that might focus on a single type of knowledge, their taxonomy covers a broad spectrum of knowledge types (from ontologies and graphs to equations and constraints) and integration methods, offering a comprehensive view of techniques for enhancing DL with domain knowledge. Although these surveys on informed or knowledge-augmented ML illustrate the landscape of how expert knowledge can influence learning algorithms, they focus primarily on the theoretical aspects of ML rather than on the domain-specific contexts of knowledge integration practices employed by domain practitioners.

Another line of surveys provides comprehensive overviews of NeSyAI, laying the groundwork for bridging symbolic reasoning and sub-symbolic perception. For example, Besold *et al*. present a broad survey of NeSyAI systems, categorizing loose and tight integration strategies for combining symbolic components with ANNs [75]. More recent NeSyAI surveys continue this direction, highlighting how hybrid methods leverage the reasoning capabilities of symbolic representations alongside the robust learning capabilities of DNNs [76], [77]. These works underscore the benefits of explicit knowledge representations, such as logical rules or grounded propositions, in improving the interpretability and generalizability of AI systems. However, NeSyAI surveys largely focus on classical, abstract AI tasks (*e.g*., image recognition, knowledge base reasoning), as introduced in the preceding subsection (Sec. II-C), rather than examining how domain practitioners inject knowledge into AI for real-world, domain-specific problems.

Domain-specific survey studies also exist that document how AI techniques are enhanced with domain knowledge in particular fields. In the medical and biomedical domains, for instance, Xie *et al*. survey approaches for incorporating medical domain knowledge into DL models for medical tasks, including disease diagnosis, lesion detection, and treatment recommendation [78]. They observe that domain knowledge in healthcare (*e.g*., clinical ontologies, anatomical constraints, and expert-defined features) can improve both the data efficiency and trustworthiness of DNNs in practice. Similarly, Cai *et al*. review methods for integrating biomedical knowledge into Natural Language Processing (NLP) models for text analysis, highlighting techniques such as medical ontologies for knowledge structure enrichment in LLM contexts [79]. In physical science domains, Meng *et al*. present a comprehensive survey of physics-informed machine learning, summarizing how fundamental physics laws, expressed as equations, simulation data, symmetry constraints, *etc*., are used to guide model training [80]. Domain-driven surveys such as these underscore a recurring finding: incorporating expert knowledge (whether clinical knowledge, physical laws, or other domain principles) tends to improve model robustness, interpretability, and sample efficiency. However, each such survey remains confined to its specific domain or knowledge modality and does not generalize beyond its particular context. The commonalities that exist across these domain-specific efforts are not immediately clear, which represents one of the gaps our survey aims to address by examining knowledge functions and utilization mechanisms across different domains.

The three lines of prior surveys have each illuminated pieces of the puzzle, including the theories of knowledge-enhanced ML, the techniques of NeSyAI with knowledge integration, and the case-specific practices of knowledge utilization with AI, but a holistic perspective remains missing. To our knowledge, existing surveys do not comprehensively discuss *utilizing knowledge for AI through DSRs* from a cross-domain, application-agnostic viewpoint. This survey distinguishes itself by disentangling the function of human knowledge from the mechanism of its utilization through a two-dimensional framework, and by mapping how these dimensions manifest across a wide array of domains. This approach allows us, perhaps for the first time, to connect insights from otherwise disparate research streams. For example, a formalism used to constrain legal document processing may serve a similar discriminative function as a set of logical predicates used to validate outputs in circuit design tools; these two applications would rarely be discussed together in previous surveys. By investigating such cases under a unified framework, we aim to reveal generalizable principles for computablizing human knowledge that apply across diverse domains.

E. Bibliometrics Details

We designed a systematic paper selection protocol to capture significant research regarding DSR-AI approaches. The scope of selection covers publications from premier academic venues, including flagship journals such as *Nature*, *Science*, and their affiliated series, alongside leading computer science conferences, *IEEE* and *ACM* transactions across all computer science disciplines, and high-quality journals on relevant applications such as construction engineering and bioinformatics. We pushed a deliberate focus on papers that demonstrably advance the frontiers of their respective domains, while proof-of-concept works on classical AI tasks that make general and superficial treatments regarding DSR-AI approaches are considered out of the scope of this survey, which is consistent with the difference discussed in Sec. II-C.

We leveraged keyword-based filtering encompassing: (i) broader and adjacent concepts relevant to the DSR scope (*e.g*., "knowledge representation", "knowledge engineering"); (ii) terms denoting the artifact families included in our definition (*i.e*., "data scheme", "notation system", "formal grammar", "declarative model", "domain-specific language"); and (iii) associated technical constructs [1] related to instantiations of DSR

[1]In the context of DSR, a construct is a fundamental building block used to represent core concepts, entities, operations, or relationships within a specialized field. Constructs provide the atomic units with inherent domain-specific meaning and representation, which can be composed to model complex systems or knowledge within that domain.

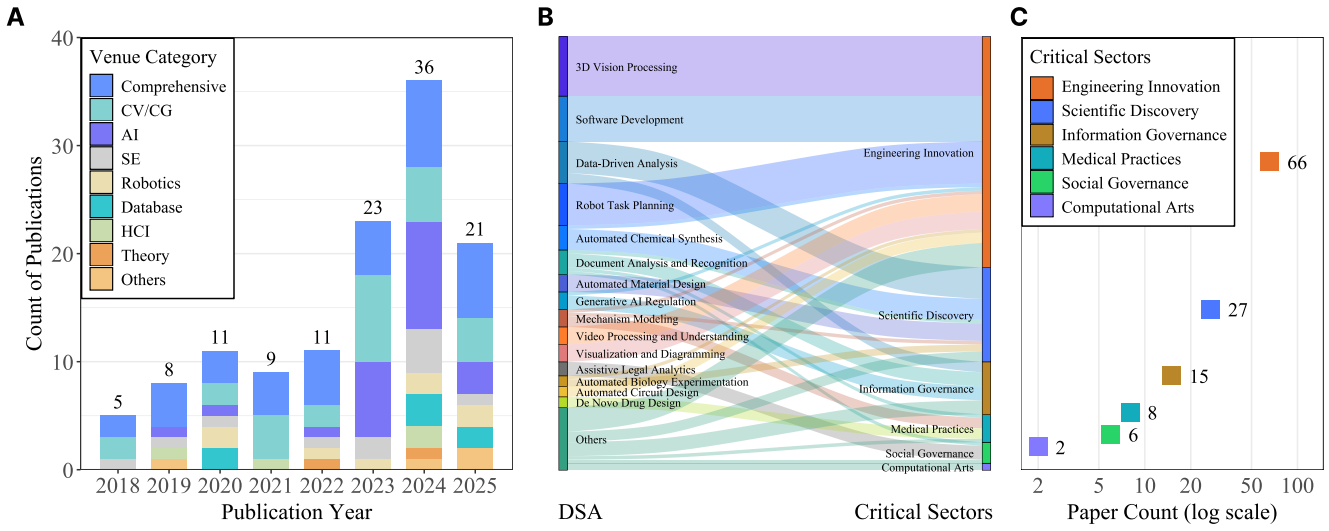


Fig. 2. **Bibliometrics details regarding DSR-AI literature across temporal, application, and sectoral dimensions. (A)** Annual publication counts from 2018 to 2025 (cut-off by August 31st), with the x-axis representing publication year and the y-axis representing paper counts; data points and bars are color-coded by Venue Category. **(B)** Alluvial diagram illustrating the relationships between Domain-Specific Applications **(DSA, left panel)** and Critical Sectors **(right panel)**; stream widths are proportional to the number of papers within each DSA–sector pairing, while node widths reflect marginal totals across categories. **(C)** Aggregate paper counts by Critical Sector over the study period (2018–2025).

(*e.g.*, "context-free grammar", "logic programming", "program synthesis", "constraint representation", *etc.*). Notably, these keywords, particularly those at the third level, were not directly proposed at the initial stage of the paper selection procedure; rather, they were iteratively refined through a snowballing sampling process over the citation network [81].

Through iterative application of these source, style, and terminological criteria, we consolidated a final corpus of 124 seminal publications, from the initial attempts of DSR-AI approaches around 2018, including the marked growth around 2023 that likely coincided with the success of LLMs. Each selected paper was analyzed following the PRISMA methodology [82]. We grouped the publication venues into 9 categories, with the most common being Comprehensive (publications in flagship journals presenting interdisciplinary approaches), representing 38 papers (30.6%), followed by CV/CG (Computer Vision and Graphics) with 27 papers (21.8%), and AI with 23 papers (18.5%) (see Fig. 2A). The domain-specific applications covered 27 unique categories, with the top 15 applications making up 85.5 % of all publications, while the remaining 12 are distributed across 14.5% additional applications (see Fig. 2B). Further, our analysis recalled the 6 distinct critical sectors, with Engineering Innovation dominating the landscape (66 papers, 53.2%), followed by Scientific Discovery (27 papers, 21.8%), and Information Governance (15 papers, 12.1%) (see Fig. 2C).

III. Framework and Taxonomy

This section begins by examining the cognitive basis of human knowledge, providing an overview of several parallel cognitive theories and focusing on Bayesian cognitive science as the theoretical foundation of the framework (Sec. III-A). We then introduce the framework for disentangling domain context and the methodology for knowledge-AI incorporation (Sec. III-B). Subsequently, we expand the taxonomy of the framework along its two dimensions: (i) the functions of human knowledge (Sec. III-C); and (ii) the mechanisms by which human knowledge is utilized in AI (Sec. III-D), respectively (see Fig. 3 for visual illustration).

A. The Cognitive Basis of Human Knowledge

Human knowledge emerges from intricate cognitive architectures that have been examined through multiple theoretical lenses. Philosophically, knowledge is often partitioned into declarative knowledge (facts, concepts, and propositions about the world) and procedural knowledge (skills and methods for executing tasks). While this dichotomy highlights distinctions between "knowing that" and "knowing how", it is not fully Mutually Exclusive and Collectively Exhaustive (MECE), as many real-world competencies—such as social intuition or creative problem-solving—blur these boundaries and remain subjects of ongoing debate regarding their classification and neural substrates [83].

From a cognitive neuroscience perspective, knowledge is categorized by acquisition source: language-derived knowledge (abstract representations acquired symbolically) and sensory-derived knowledge (embodied representations grounded in perception). This framework emphasizes fundamental brain mechanisms, whereby language-based knowledge engages prefrontal and temporal regions, while sensory-derived knowledge relies on modality-specific cortical areas. However, this view primarily addresses elementary perceptual-cognitive interfaces and less explicitly accounts for higher-order integrative processes such as metacognition [84].

Complementary perspectives also include distinctions between episodic and semantic memory [85], where episodic knowledge binds contextual details to lived experiences, and

semantic knowledge generalizes across contexts. Similarly, implicit versus explicit knowledge frameworks differentiate unconscious, automatized skills (*e.g.*, motor coordination) from consciously accessible facts, revealing dissociable neural pathways [86]. Despite their theoretical merit, these taxonomies exhibit overlaps and contextual dependencies.

Among these parallel theories of human knowledge, given our focus on domain-specific applications with AI, we adopt a pragmatic stance centered on operationalizing knowledge for computational systems. We therefore select the most fundamental theory that aims to describe and handle the relations between the use of human knowledge and empirical observation, top-down derivation and bottom-up induction, human intelligence and computational power—Bayesian theory. Bayesian theory provides this *lingua franca* through probabilistic frameworks that model cognition as rational inference under uncertainty [87]. By treating task objective as probabilistic beliefs $p(h \mid e)$ updated via prior hypothesis $p(h)$ and likelihood of evidence $p(e \mid h)$, formulated as

$$\underbrace{p(h \mid e)}_{\text{posterior}} \propto \underbrace{p(h)}_{\text{prior}} \underbrace{p(e \mid h)}_{\text{likelihood}} . \tag{1}$$

It aligns human and AI within a common computational paradigm. Under the Bayesian cognitive science perspective, which posits that the mind approximates optimal probabilistic inference, we classify human knowledge according to its function in probabilistic computation.

a) Generative function of knowledge: The generative function of human knowledge posits a latent variable as the generator of an entire system within a limited scope, mirroring the system itself, analogous to a "digital twin" of the system. This function assumes latent variables z that reflect the underlying principles governing system dynamics, thereby enabling the joint distribution $p(x, z)$, whose mathematical significance lies in the complete knowledge of how system states and observations co-occur, to synthesize observable outputs x of the system as it evolves under the governing dynamics. Formally, this represents a prior over mechanisms:

$$\underbrace{p(x, z)}_{\text{system simulation}} = \underbrace{p(z)}_{\text{knowledge}} \underbrace{p(x \mid z)}_{\text{generator}} . \tag{2}$$

This function enables the utilization of knowledge to simulate or generate novel, plausible observations of the system, embodying the capacity to imagine or predict sensory data based on internal models. Among the cases illustrated in Sec. I, the knowledge of viral mutation mechanisms exemplifies this generative function of knowledge [16].

b) Discriminative function of knowledge: The discriminative function of human knowledge presupposes a classifier operating within a limited scope according to specific criteria. This function directs knowledge toward directly mapping observations to inferences or decisions about underlying states, circumventing the need for explicit modeling of data generation. It assumes conditional probabilities $p(z \mid x)$ that reflect the judgments rendered by knowledge z concerning observations x. This function employs Bayes' equation for posterior probability:

$$\underbrace{p(z \mid x)}_{\text{discriminator}} = \underbrace{p(x, z)}_{\text{system}} / \underbrace{p(x)}_{\text{observation}} , \tag{3}$$

where $p(x, z)$ represents the system generating the observations. This function enables the application of knowledge to render judgments about specific system observations. Among the cases illustrated in Sec. I, the knowledge assessing whether AI-generated motions conform to physical principles exemplifies this discriminative function of knowledge [17].

c) Descriptive function of knowledge: The descriptive function of human knowledge presupposes a structure, format, or layout governing the input or output of the application task. This function focuses knowledge on addressing the observable space rather than the internal system dynamics (generative function) or the rules for evaluating the system (discriminative function). It assumes a description space $\mathcal{Z}$ designated by human knowledge that specifies the observable space, where x and x' represent observations before and after the application of descriptor z, respectively. This function employs the Bayesian factorization rule:

$$\underbrace{p(x' \mid x)}_{\text{descriptor-constrained observation}} = \sum_{z \in \mathcal{Z}} \underbrace{p(x' \mid z)}_{\text{descriptor}} \underbrace{p(z \mid x)}_{\text{receiver}} . \tag{4}$$

This function enables the application of knowledge to constrain the observable spaces of the system to satisfy specific requirements or adapt to broader environments. Among the cases illustrated in Sec. I, the knowledge required to format NL-based experimental procedures extracted from documents as machine-executable programming languages exemplifies this descriptive function of knowledge [18].

This tripartite division, drawing inspiration from Bayesian cognitive science, provides a cognitively grounded foundation for our proposed framework.

B. The Framework for Disentanglement

The persistent fragmentation in DSR-AI research stems from an inherent entanglement between *the domain knowledge* is being leveraged and *the methods* by which it is computationally utilized within AI solutions. When domain practitioners craft DSRs, they implicitly fuse specific properties of their knowledge (*e.g.*, virology mutation rules or experimental execution constraints) with methodological choices for integrating that knowledge into AI workflows (*e.g.*, symbolic search-space construction or logic-based validation). This conflation results in three critical limitations: (i) reinvention overhead—isolated teams repeatedly rediscover integration patterns (*e.g.*, constraint programming) for each new application domain, resulting in "reinventing the wheel"-style wasted resources; (ii) transferability barriers—solutions become siloed, such that a physics-informed neural network for fluid dynamics cannot leverage methodologies from structurally similar materials science DSRs; and (iii) democratization gaps—domain practitioners without AI expertise struggle to adapt solutions, as context-specific implementations obscure reusable principles.

To address this, we introduce a structured framework that explicitly spans two fundamental dimensions:

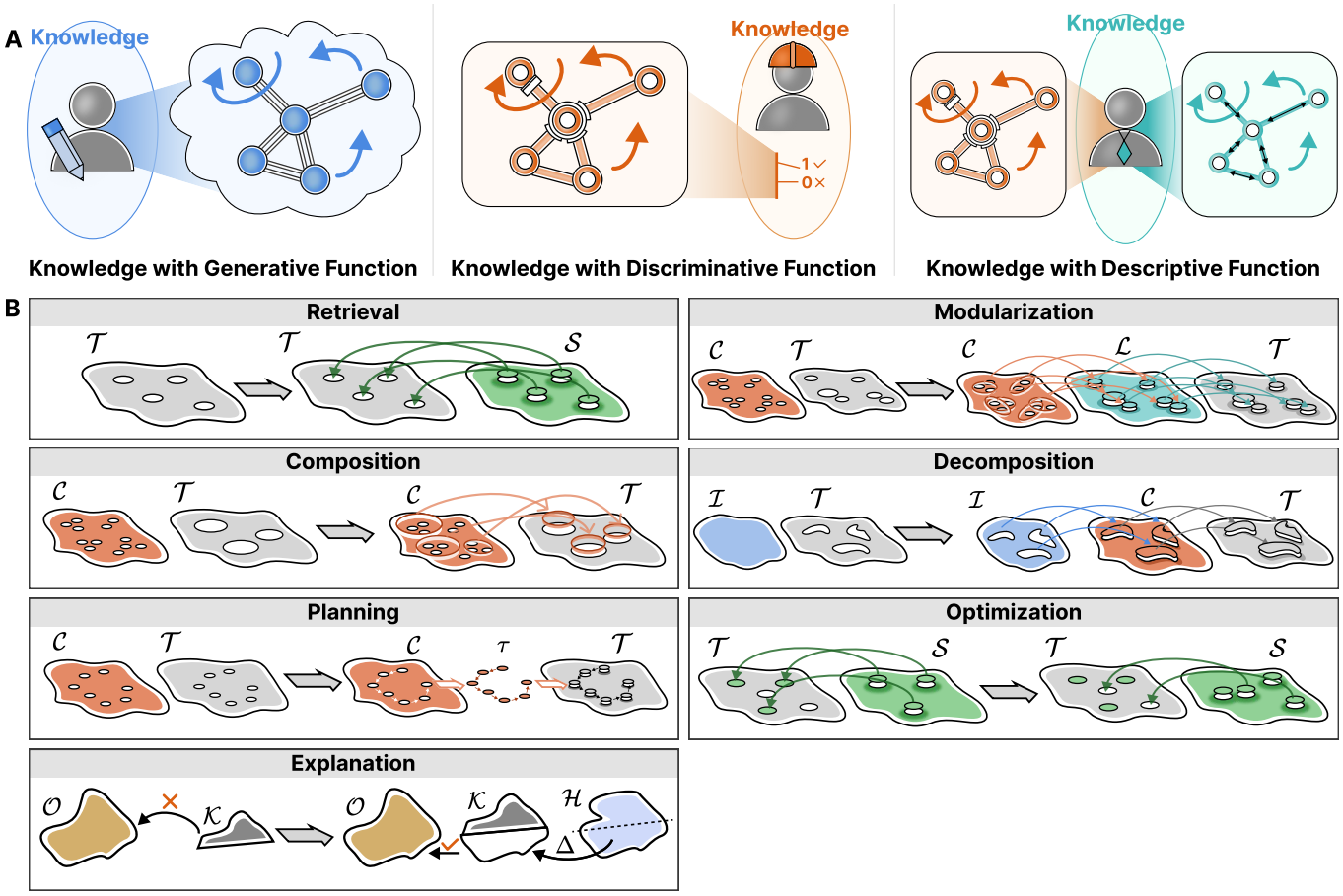


Fig. 3. **Intuitive, abstraction illustration of the two-dimensional framework. (A)** The dimension of knowledge function, including three categories: generative, discriminative, and descriptive functions. **(B)** The dimension of knowledge utilization mechanism, including seven categories: retrieval, modularization, composition, decomposition, planning, optimization, and explanation. Notably, **(A)** and **(B)** do not share the same color coding scheme.

a) The functions of human knowledge: This dimension addresses the question *"What property of the knowledge is represented?"*, classifying the knowledge being encoded in DSRs according to its probabilistic function: generative, discriminative, and descriptive. This dimension focuses solely on the functions of human knowledge themselves, independent of how they are utilized in DSR-AI approaches. The taxonomies for this dimension are detailed in Sec. III-C.

b) The mechanisms of AI knowledge utilization: This dimension addresses the question *"How the knowledge is computed with AI?"*, codifying how AI methods computationally leverage DSRs to solve tasks. It is grounded in the fundamental utilities of AI systems from the classical perspective of treating AI as a search for solutions to problems: retrieval, modularization, composition, decomposition, planning, optimization, and explanation. This dimension focuses exclusively on how knowledge is utilized in DSR-AI approaches, regardless of the specific functions it constitutes. The taxonomies for this dimension are detailed in Sec. III-D.

Within this framework, knowledge with a specific function (*e.g.*, generative biomechanical laws) can be utilized through multiple mechanisms (*e.g.*, composition in prosthetics design or planning in surgical robotics). Conversely, a knowledge utilization mechanism (*e.g.*, retrieval) can typically be applied across various knowledge functions (*e.g.*, discriminative safety rules in autonomous vehicles or descriptive protocols in chemical synthesis). In a particular DSR-AI approach, practitioners may integrate knowledge spanning multiple functions, and apply multiple methodologies for knowledge utilization at different stages within the overall workflow. These two dimensions enable the integration of originally fragmented application cases under a standardized, unified narrative system, transcending their superficial appearances to reveal their cognitive and computational essence, thereby precisely positioning each DSR-AI approach within the global landscape.

By spanning these two dimensions, our framework establishes a unified taxonomy for dissecting, comparing, and creatively developing DSR-AI approaches. It provides the foundation for shifting discourse from isolated case studies to a principled practice of knowledge-integrated AI solutions, revealing shared design choices and computational patterns across domains while respecting their contextual uniqueness. This disentanglement is pivotal for scaling AI democratization—enabling domain experts to specify knowledge contexts independently of implementing integration mechanics, thereby lowering barriers to cross-domain innovation.

C. The Dimension of Knowledge Function

This dimension abstracts domain knowledge properties to classify knowledge according to its fundamental function within a given domain. Grounded in Bayesian cognitive theory, it distills how human expertise shapes the interaction between knowledge and reality, whether through simulating system mechanisms, evaluating system states, or formatting observable spaces. Notably, these functions of human knowledge operate independently of their utilization by AI (see Fig. 3A).

a) Knowledge of generative function: Knowledge of generative function operates as a simulator, encoding latent variables that mirror domain mechanisms to synthesize observations. It constructs joint probability distributions $p(x, z)$, where domain principles $p(z)$ govern the generation of observable outputs $p(x \mid z)$. In the mechanism simulation of viral escape, for instance, knowledge of visual mutation behavior is precisely captured by a DSR symbolically instantiated through a self-contained production set [16]. Crucially, such knowledge function serves as a "computational universe"—a self-contained digital twin within a limited scope that synthesizes predictive and counterfactual system states while respecting domain laws. Its power lies not merely in prediction but in enabling exploratory inference: AI can probe "what-if" questions unpractical to test empirically.

b) Knowledge of discriminative function: Knowledge of discriminative function serves as a validator, focusing on conditional relationships $p(z \mid x)$ to judge observations against domain logic and constraints. It bypasses generative modeling to directly evaluate system states, acting as a domain-specific auditor that distinguishes valid from invalid configurations. In motion graphics animation, for example, the knowledge of physical kinematics and dynamics governing joint articulation behavior is represented by a DSR instantiated through a knowledge base consisting of logic programs, accompanied an ILP-based reasoning module [17]. Such knowledge function excels in rendering rapid binary verdicts (*e.g.*, physical plausibility or implausibility), making it indispensable for real-time applications where false positives may carry severe risks.

c) Knowledge of descriptive function: Knowledge of descriptive function acts as a structural scaffold, defining the syntax and semantics of observable spaces $\mathcal{Z}$. It abstracts away underlying mechanisms to focus purely on input-output formatting, such as translating between human-interpretable concepts and machine-executable representations. In automated chemical synthesis, knowledge of the hardware devices in the target self-driving laboratory is incorporated into a DSR instantiated through a markup programming language with a compiler to translate input control flow into execution instructions for the devices [18]. Such knowledge function standardizes heterogeneous information and ensures computational interoperability across subsystems, thereby serving as the "communication bridge" between information from different knowledge sources.

This dimension thus reveals that functions of human knowledge are application-invariant computational primitives. By classifying DSRs through this lens, we unlock cross-disciplinary understanding of human knowledge.

D. The Dimension of Knowledge Utilization Mechanism

This dimension systematizes the algorithmic orchestration of knowledge within AI workflows. Unlike the knowledge function dimension, which addresses "what" is represented, this dimension focuses exclusively on "how" knowledge becomes computable, revealing transferable computational mechanisms that recur across domains. We present each mechanism following the objective function description, which is standard practice in AI literature, with definitions of input, output, and their respective spaces to enhance readability. Notably, these mechanisms of knowledge utilization are independent of the intrinsic functions of the knowledge (see Fig. 3B).

a) Retrieval: The input to retrieval is the requirement for an instance solution $s \in \mathcal{S}$ to complete a specific set of tasks $t \in \mathcal{T}$, and its corresponding space $\mathcal{S}$ is the full set of candidate instance solutions. The output is the solution that best satisfies the requirements of t. Therefore, retrieval can be formalized as

$$s^* = \arg\max_{s \in \mathcal{S}} \sum_{t \in \mathcal{T}} f(s \mid t)\,, \tag{5}$$

where $f : \mathcal{T} \times \mathcal{S} \to \mathbb{R}$ is a function that quantifies the fitness of a candidate solution s with respect to task t. Knowledge with different functions may play distinct roles in retrieval. Knowledge with generative function can serve as a heuristics for searching for the solution, *e.g.*, querying 3D models in a gallery based on specific requirements, using heuristics derived from a DSR instantiated through functional modeling programs [88]. Knowledge with discriminative function can define the criteria for evaluating solutions, *e.g.*, retrieving similar judicial cases, judged by predicates devised through a DSR instantiated via a CFG [89]. Knowledge with descriptive function may act as structuring or formatting of the solution space for efficient querying, *e.g.*, providing references for code completion, where the reference repository is structured by a DSR instantiated by the corresponding DSL syntax [90].

b) Modularization: The input to modularization is a system or a set of knowledge with multiple atomic-level components $c \in \mathcal{C}$ for a specific set of tasks $t \in \mathcal{T}$, and its corresponding space $\mathcal{C}$ is the full set of atomic components within the system. The output is an abstract representation of the system $\mathcal{L} \subseteq \mathcal{C}^*$ [2] consisting of encapsulated abstractions of the atomic components $\{l \mid c^* \mapsto l\}$, *i.e.*, reusable functions built over the atomic components, serving as a component library that solves the target task with minimum complexity, namely, by calling the simplest and fewest functions of the abstract representation. Therefore, modularization can be formalized as

$$\mathcal{L}^* = \arg\max_{\mathcal{L} \subseteq \mathcal{C}^*} \sum_{t \in \mathcal{T}} \mathbf{1}\left(\min_{S \subseteq \mathcal{L} \text{ solves } t} |S| \right)\,, \tag{6}$$

where $|S|$ denotes the complexity of the minimal set of abstractions S successfully solving t and $\mathbf{1}(\cdot)$ denotes the indicator function. Knowledge with different functions can

[2] $\mathcal{C}^*$ denotes the production set of $\mathcal{C}$ in this context, which should be differentiated from those with star notation on the left-hand side within the equations denoting optimality.

all serve as materials and guidance for modularization, with different usage contexts. For example, although all are used for making reusable assemblies regarding Computer-Aided Design (CAD) modeling to facilitate efficiency, knowledge with generative function directly creates a DSR instantiated by assemblies of CAD programs for modeling prefabricated components [91]; knowledge with discriminative function creates a DSR instantiated by high-level encapsulations of primitive physical constraints for efficient structure validation [92]; and knowledge with descriptive function creates a DSR instantiated by reusable macros that structuralize the space of CAD code generation to reduce meaningless results [93].

c) Composition: The input to composition is the same as the input to modularization, including atomic-level components $c \in \mathcal{C}$ and tasks $t \in \mathcal{T}$. The output is an appropriate set $S \subseteq \mathcal{C}^*$, a combination of the atomic components, that best solves t. Therefore, composition can be formalized as

$$S^* = \arg\max_{S \subseteq \mathcal{C}^*} \sum_{t \in \mathcal{T}} f(S \mid t)\,, \tag{7}$$

where $f : \mathcal{T} \times \mathcal{C}^* \to \mathbb{R}$ is a function that quantifies the fitness of a composed set S with respect to task t. Knowledge with different functions can all serve as materials and guidance for composition, with different usage contexts. Knowledge with generative function, for example in AI music composition, can create a DSR instantiated through structural vectors capturing music structures to assemble larger sequences of notes [94]. Knowledge with discriminative function, for example in novel material design, can create a DSR instantiated through regular expressions that encode verified rules for constraint programming to enhance human-in-the-loop design [95]. Knowledge with descriptive function, for example in protein structure analysis, can create a DSR instantiated through primitive syntax that captures macro and micro patterns of proteins within their molecular string representation [96].

d) Decomposition: The input to decomposition is a system or a set of knowledge itself $I \in \mathcal{I}$, along with a specific set of tasks $t \in \mathcal{T}$ to solve. The output is an appropriate set consisting of atomic-level components $\{c \mid c \in \mathcal{C}, I \mapsto c\}$ that together mirror the original entire instance and maximize the coverage of solvable $t \in \mathcal{T}$. Therefore, decomposition can be formalized as

$$\mathcal{C}^* = \arg\max_{\mathcal{C}} \sum_{t \in \mathcal{T}} \mathbf{1}(S \subseteq \mathcal{C} \text{ solves } t)\,, \tag{8}$$

where $\mathbf{1}(\cdot)$ denotes the indicator function and S represents a subset selected from the resulting atomic set. Knowledge with different functions can all serve as guidance for decomposition, with different usage contexts. For instance, knowledge with generative function creates a DSR instantiated through attribute grammar to decompose real-world objects in visual perception into parts that are ready for interaction [97]. Knowledge with discriminative function creates a DSR instantiated by probabilistic graph syntax to decompose holistic software user behavior into parametric states w.r.t. transitions that are ready to be checked and verified [98]. Knowledge with descriptive function creates a DSR instantiated through a DSL to decompose the ambiguous process of programming language design into a structural, partially-quantitative workflow to facilitate user-centered design [99].

e) Planning: The input to planning is the same as the input to composition, including atomic-level components $c \in \mathcal{C}$ and tasks $t \in \mathcal{T}$. The output is an ordered set $\{\tau \mid \mathcal{C}^*, \preceq\}$, also a combination of atomic components but with a specific (partial) order restricted by both the properties of the atomic components and task requirements, that best solves t. Therefore, planning can be formalized as

$$\tau^* = \arg\max_{\tau \subseteq \mathcal{C}^*} \sum_{t \in \mathcal{T}} f(\tau \mid t)\,, \tag{9}$$

where $f : \mathcal{T} \times \mathcal{C}^* \to \mathbb{R}$ is a function that quantifies the fitness of a resulting plan τ with respect to task t. Knowledge with different functions can all serve as materials and guidance for planning, with different usage contexts. For example, knowledge with generative function can create a DSR instantiated by temporal and-or-graph grammar to synthesize robot action primitives into executable plans [100]. Knowledge with discriminative function can create a DSR instantiated by graph operator grammar as a sequential controller of AI-generated chemical retrosynthesis trajectories [101]. Knowledge with descriptive function can create a DSR instantiated by graph search structure to facilitate efficient troubleshooting and the synthesis of repair plans [102].

f) Optimization: The input to optimization is an already-existing solution $\hat{s} \in \mathcal{S}$ that solves $t \in \mathcal{T}$ but may not be sufficiently desired. The output is the solution that best satisfies the requirements of t. Therefore, optimization can be formalized as

$$s^* = \arg\max_{s \in \mathcal{S}, s \neq \hat{s}} \sum_{t \in \mathcal{T}} f(s \mid t, \hat{s})\,, \tag{10}$$

where $f : \mathcal{T} \times \mathcal{S} \to \mathbb{R}$ is a function that quantifies the fitness of a candidate solution s with respect to task t. Knowledge with different functions may play different roles in optimization. Knowledge with generative function may serve as hints for potential improvement directions of the solution, *e.g.*, iteratively updating domain-specific LLMs according to feedback generated by a DSR instantiated through structural templates [103]. Knowledge with discriminative function may be used to restrict the plausible scope for controllable optimization, *e.g.*, constraining the training process of physical dynamics learning with a DSR instantiated by structured priors encoded in ANN architecture and hyperparameters [104]. Knowledge with descriptive function may act as structuring or formatting of the solution space for efficient search for better solutions, *e.g.*, providing anchors for code manipulation, where the high-level code-block structure is represented by a DSR instantiated through syntax templates of the structures of relevant coding tasks [105].

g) Explanation: The input to explanation is observation $O \in \mathcal{O}$ and a set of background knowledge of the specific task $K \in \mathcal{K}$, especially where K may not entail the given O. The output is to find the explanation $\Delta \in \mathcal{H}$ from the hypothesis space $\mathcal{H} \subseteq \mathcal{K}^*$, which originates from the production of background knowledge, that best complements K to entail O.

Therefore, explanation can be formalized as

$$\Delta^* = \arg\max_{\Delta \in \mathcal{H}} \sum_{O \in \mathcal{O}} \mathbf{1}(K \cup \Delta \text{ entails } O), \tag{11}$$

where $\mathbf{1}(\cdot)$ denotes the indicator function and $K \cup \Delta$ represents the background knowledge complemented with the explanation. Knowledge with different functions may play different roles in explanation. Knowledge with generative function may directly model the hypothesis space for explanation, *e.g*., synthesizing programs from records of human languages by a DSR instantiated through Probabilistic Programming Language (PPL) to verify the morpho-phonology properties of human language [106]. Knowledge with discriminative function may represent the criteria for judging the entailment, *e.g*., handling molecular data with a DSR instantiated through logic programs of a pathway knowledge base to evaluate whether the patterns extracted from the data are consistent with the pathway knowledge [107]. Knowledge with descriptive function may structure the space for explanation searching and synthesis, *e.g*., creating a DSR instantiated through a DSL to provide programmatic summarization of hypothesis testing within geospatial data analysis coding [108].

This dimension also reveals that methodologies for AI utilizing human knowledge are application-general computational practices. Through this lens, we unlock a computer-scientist perspective on knowledge.

IV. Analysis on DSR-AI Approaches

This section begins with an overview of the literature landscape based on the two-dimensional framework (Sec. IV-A). Following this organization, we examine three primary groups of design trends (Secs. IV-B to IV-D), and additional local design patterns (Sec. IV-E). Subsequently, we synthesize the lessons derived from this analysis into practical principles for future DSR-AI approach development (Sec. IV-F).

A. Overview of the Landscape

Through a comprehensive literature review of the selected papers using the two-dimensional framework, which encompasses a sophisticated indicator space, a distinctive landscape of DSR-AI approaches emerges. Three primarily grouped design trends emerge. First, 21% of the works leverage knowledge with generative function under composition mechanism, *i.e*., Generative-Composition approaches. Within this group, 34.6% of the works incorporate with other knowledge functions and utilization mechanisms, including Generative-Modularization/Decomposition (23.1%), Discriminative-Optimization (15.4%), and Discriminative-Modularization (7.7%). Second, 16.9% of the works leverage knowledge with discriminative function under explanation mechanism, *i.e*., Discriminative-Explanation approaches. Within this group, 28.6% of the works incorporate with other knowledge functions and utilization mechanisms, including Discriminative-Optimization (19%), Discriminative-Modularization/Decomposition (28.6%), and Discriminative-Composition (14.3%). Third, 15.3% of the works leverage knowledge with descriptive function under retrieval mechanism, *i.e*., Descriptive-Retrieval approaches. Within this group, 73.7% of the works incorporate with other knowledge functions and utilization mechanisms, including Discriminative-Optimization (26.3%), Descriptive-Optimization (10.5%), Descriptive-Composition (26.3%), and Descriptive-Planning (15.8%). Besides, there are less widespread yet characteristic local practice trends, such as Descriptive-Composition (15.3%) and Generative-Explanation (4%). The detailed landscape of literature, as delineated by the two-dimensional framework, is presented in Fig. 4. In the following subsections, we examine these design trends to illuminate the design rationales that inform DSR-AI approach development.

B. Group 1: Generative-Composition Approaches

The first significant aggregative design trend is incorporating generative knowledge under composition mechanism, *i.e*., Generative-Composition approaches, relevant to tasks characterized by *invention* and *creation*, including design, synthesis, and constructive endeavors.

a) Pure Generative-Composition approaches: Knowledge with generative function and composition mechanisms naturally align in their conceptual foundations. The former acts as the "first principles" of a physical or knowledge system, reflecting the underlying logic that governs how the system behaves and evolves. The latter assembles these first principles into coherent systems, transforming a set of primitives into a substantially larger and more complex set. This synergy explains their co-occurrence. To illustrate how specific applications can abstract to this function-mechanism combination, consider Hie *et al*.'s [16] approach to viral immune escape prediction. Starting from the concrete challenge of predicting how viruses evolve to evade immune responses, the authors observed that viral mutations follow systematic evolutionary principles requiring the composition of protein sequences into escape scenarios; They abstracted this observation into generative knowledge, *i.e*., a protein language model capturing the "grammar" of amino-acid substitution patterns, and operationalized it through composition, *i.e*., assembling mutations of protein sequences into escape scenarios by learning from whole-genome viral sequences.

In Generative-Composition systems across design-centric domains, the DSR defines generative primitives, and an accompanying learning or search routine composes them into higher-order structures. In 3D vision processing, Jiang *et al*. [109] encoded scene regularities as stochastic grammar rules to synthesize configurable 3D scenes. Similarly, in computational arts, Huang *et al*. [94] represented the generative knowledge of long-range repetitive harmonic structure to generate coherent music. For *de novo* drug design, Yang *et al*. [110] modelled ensembles of RNA conformers *in vivo* to compose alternative structural states for regulation explaining. Bozhüyük *et al*. [114] encoded modularity rules of non-ribosomal peptide synthetases to compose chimeric fragments for new biosynthetic pathways. Qu and Krishnapriyan [115] composed multi-head neighbour interactions to scale inter-atomic potentials. For numerical optimization, Lai *et al*. [112]

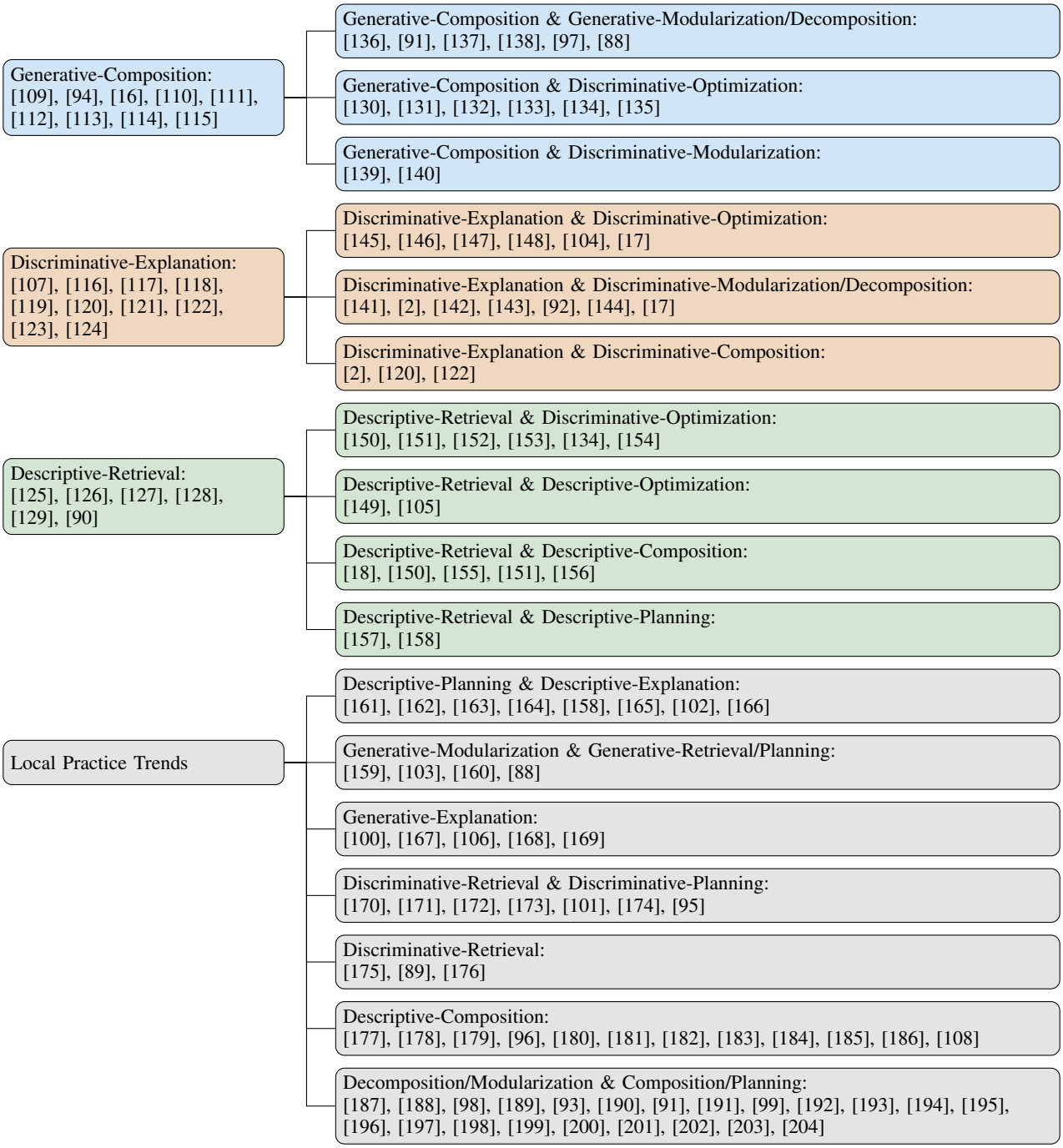


Fig. 4. **The distribution of literature under the analysis by the two-dimensional framework**

composed physics-driven priors as algorithmic blocks to compose interpretable solvers. Building on these applications in specific domains, Chen *et al*. [113] generalized the paradigm by composing expert-specific generators into an aggregated multi-expert model for cross-domain problem-solving.

b) Generative-Composition with decomposition and modularization mechanisms: Generative-Composition approaches are often incorporated in parallel alongside modularization and decomposition mechanisms. This hybrid approach extends beyond pure generation by using the same generative knowledge to first decompose systems into atomic components or modularize them into reusable libraries. For example, Yuan *et al*. [138] addressed the challenge of enabling LLMs to both understand and compose music in a theory-grounded way. They observed that musical compositions follow systematic theoretical principles, requiring decomposition into manageable primitives for processing. Accordingly, they abstracted this observation to encode music-theoretic principles—including pitch, duration, chord, and form relationships—as generative knowledge, and implemented three complementary mechanisms: composition (assembling musical elements via token generation), decompo-

sition (breaking musical pieces into theory-grounded primitives), and modularization (organizing musical elements into hierarchical score structures).

Building on the integration of generative function with decomposition and modularization, a line of domain-specific works shows how generative knowledge guides both the breakdown and recombination of complex systems. In visualization, Ye *et al*. [136] encoded mathematical objects and relations as a declarative DSL with constraint-based layouts for precise and style-consistent visualizations. Li and Yuan [137] decomposed 2D visual designs into encoding primitives, and composed them under constraints to formalize a grammar of tree layouts. Jones *et al*. [91] learned the generative building blocks of 3D shape modeling programs by decomposing program traces and recomposing them to synthesize or edit shapes. Mandi *et al*. [97] extended this to visualization program synthesis, decomposing RGB images into joint hypotheses and composing executable, physically plausible structures.

c) Generative-Composition with Discriminative-Optimization approaches: There are also hybrid approaches, such as Generative-Composition with Discriminative-Optimization, that jointly exploit different function-mechanism pairs. These approaches combine function-mechanism pairs in a propose-constrain framework, where the Generative-Composition component explores possible creations, while the Discriminative-Optimization component constrains this search to plausible outcomes. For example, considering Zhao *et al*.'s [130] work on legged robot design. They observed that robot's body structure follows systematic assembly rules while requiring terrain-specific performance optimization. They abstracted this observation into generative knowledge (graph grammar encoding kinematic feasibility and connection rules) and discriminative knowledge (physics simulator and terrain stability predicates), then composed robot bodies via grammar derivations and optimized controller parameters to maximize terrain performance.

Building on this general framework, a variety of domain-specific works demonstrate the synergy of generative and discriminative mechanisms. In synthetic biology, Ai *et al*. [133] treated metabolic networks as generative knowledge, and assay outcomes as discriminative signals, composing gene queries and optimizing for human-comprehensible information gain. In a similar vein, in computer program synthesis, Alrashedy *et al*. [135] employed the CAD DSL as generative knowledge and executable feedback as discriminative guidance, composing chained Vision-Language Model (VLM)-generated primitives and optimizing prompts iteratively to repair code. Yao *et al*. [134] synthesized functionally correct HDL code by composing bugfix patterns with an LLM, and optimizing design through execution-in-the-loop verification. Wang *et al*. [132] generalized program synthesis across DSLs spanning diverse domains by composing programs via grammar-based decoding, and optimizing with constrained greedy search. Merrell [131] targeted the generation of geometric shapes by composing and optimizing structures under planar and intersection constraints to balance style and plausibility.

d) Generative-Composition with Discriminative-Modularization approaches: Additionally, some approaches integrate Generative-Composition with Discriminative-Modularization, using knowledge with discriminative functions to manage large and complex exploration spaces where generative principles alone are insufficient. As a concrete example of how this function-mechanism combination is instantiated in practice, consider Mavrogiannis *et al*.'s [139] approach to robot task planning from natural-language cooking recipes. Starting from the challenge of translating human cooking instructions into executable robot behaviors, they observed that atomic cooking actions follow systematic composition rules, while their validation and organization depend on semantic understanding and executability constraints. Accordingly, they formulated a modularized library of atomic actions with pre/post conditions as generative primitives, and incorporated discriminative knowledge through cooking procedure checks. These actions were then composed into temporal cooking logic specifications to ensure semantic validity. Similarly, in integrated circuit design, Liu *et al*. [140] encoded Register Transfer Level (RTL) construction regularities, such as connection patterns, as generative primitives. These primitives were composed to synthesize high-quality Verilog DSL code, while discriminative cues from computation graph connectivity and port consistency were used to modularize the design space into validated components with interfaces.

C. Group 2: Discriminative-Explanation Approaches

The second design trend is the incorporation of knowledge with discriminative function under explanation mechanism, *i.e*., Discriminative-Explanation approaches. Generally, approaches following this pattern tend to be relevant to tasks characterized by *analysis* and *guardrail*, including discovery, verification, and guidance operations.

a) Pure Discriminative-Explanation approaches: Knowledge with discriminative function and explanation mechanisms align naturally: discriminative knowledge serves as "second principles" to capture explicit, observable system rules, while explanation mechanism serves as the operational framework for reasoning these rules to assess system states. For example, consider Li *et al*.'s [117] work to detect and explain human actions in complex visual scenes. They observed that human activities can be characterized by constraints between body parts and interacting objects. Accordingly, they formulated discriminative knowledge in the form of action taxonomies, human-part states, and object roles as filtering criteria, and operationalized it through graph-based reasoning to generate interpretable action explanations.

Building on this conceptual alignment, many domain-specific studies demonstrate how discriminative knowledge and explanation mechanisms operate in practice. In proteomics, Babur *et al*. [107] treated protein–protein influences as discriminative knowledge to perform logic-guided causal graph construction to explain differential proteomic signals. Liu *et al*. [123] wrapped biomedical ontologies and assay constraints to produce stepwise protein explanations. In neu-

roimaging, Abdallah *et al*. [116] encoded anatomical ontologies as discriminative predicates and applied probabilistic logic programming to infer brain–behavior associations. For embodied Question Answering (QA), Tan *et al*. [118] modeled commonsense spatial-temporal relations as discriminators to generate knowledge-guided explanations. In recommendation systems, Yao *et al*. [124] applied logic-regularized reasoning over symbolic interaction rules for interpretable recommendations. In visualization, Wang *et al*. [119] encoded design semantics as discriminative constraints to trace explanations of chart recommendations, while Wang *et al*. [121] grounded LLM-elicited preferences in Draco's rule base to provide machine-checkable design templates.

b) Discriminative-Explanation with optimization mechanisms: Beyond the pure Discriminative-Explanation approaches, some approaches integrate optimization to form a comprehensive analytic loop. This combines discriminative knowledge for evaluation, explanation for diagnosis, and optimization for refinement, aligning with the fundamental "proposal-evaluation-explanation-optimization" cycle that characterizes many critical scientific and engineering processes. For a practical demonstration, consider Ma and Agrawala's [17] approach for motion graphics generation. In order to create physically plausible animations, the authors observed that animated trajectories must satisfy kinematic laws and contact constraints while requiring iterative refinement for realism. They abstracted this observation as discriminative knowledge capturing kinematic and contact plausibility, and operationalized it through explaining why LLM-generated trajectories violate physical principles, followed by execution-in-the-loop optimization to achieve verified physical correctness.

Discriminative–Explanation with Optimization approaches are common in engineering, where guardrails justify and improve decisions. In the realm of robot task transfer, Lázaro-Gredilla *et al*. [145] encoded commonsense predicates as discriminative knowledge, performing abductive reasoning and search-based optimization for zero-shot, interpretable task transfer. Echoing the same abductive idea in law, Huang *et al*. [146] modeled statutory logic as a discriminative theory, applying semi-supervised abductive reasoning with neural–symbolic optimization to improve sentencing accuracy with explanations. In vision and language, Gao *et al*. [148] encoded layout and linguistic rules as discriminative oracles to explain OCR decisions and optimize end-to-end performance. In materials design, Park *et al*. [147] applied chemistry DSL constraints to explain and optimize catalyst design under mechanistic rules. In physical system modeling, Huang *et al*. [104] cast conservation principles as discriminative constraints to explain model behavior and optimize DNN training for data-efficiency and multiphysics generalization.

c) Discriminative-Explanation with decomposition and modularization approaches: Some approaches integrate explanation, decomposition, and modularization to extend beyond Discriminative-Explanation methods, enabling systems to be broken into atomic components or reusable modules for granular analysis. For example, Zhang *et al*.'s [143] addressed the challenge of accurately segmenting human body parts in images while preserving global consistency. They observed that human bodies exhibit hierarchical part-whole structures that can guide segmentation. Based on this insight, they formulated discriminative knowledge in the form of contour-coherency rules encoded as grammar constraints to filter plausible layouts, and decomposed the human body into grammar-induced hierarchical parts to explain the segmentation.

Discriminative-Explanation approaches combining explanation with modularization or decomposition have produced concise, interpretable solutions across domains. In vision programming, Du *et al*. [141] transformed 3D meshes into editable modeling programs by encoding validity rules as discriminative knowledge, using inverse-graphics reasoning and decomposing shapes into parameterized primitives for compact, verifiable CAD models. Jones *et al*. [92] extended this to structural reuse by inducing modular libraries guided by discriminative priors for geometric regularity and conciseness. Zhang *et al*. [144] applied scene-object semantics as discriminative knowledge for action anticipation, decomposing scenes into modular components to explain outcomes. In biomedicine, Fish *et al*. [142] formalized RNA regulatory rules as discriminative knowledge, integrating motif–structure evidence and decomposing programs into exon-level modules to identify SNRPA1 as a prometastatic splicing enhancer.

d) Discriminative-Explanation with composition approaches: Some approaches integrate explanation and composition, extending basic Discriminative-Explanation frameworks by synthesizing multiple hypotheses into unified explanatory models. The composition mechanism treats undecided predicates as atomic components, systematically combining them to form comprehensive explanations. These explanations can exhibit significant complexity and demonstrate long-tail distributions characteristic of real-world observational domains. For example, Zhang *et al*. [120] addressed the challenge of detecting and explaining defects in complex manufacturing assemblies. They observed that such defects can be systematically identified using logical rules that govern component relationships and geometric constraints. Based on this insight, they formulated discriminative knowledge as bill-of-materials logic and assembly dependencies serving as filtering rules, and implemented explanation through symbolic execution to trace the causes of anomalies, together with composition via LLM-generated logic code that integrates atomic predicates into comprehensive diagnostic programs across product hierarchies. In equation discovery, Udrescu and Tegmark [2] applied compositionality, smoothness, and symmetry as discriminative criteria, generating closed-form explanations by composing primitive operators into interpretable expressions. In industrial fault analysis, Zhang *et al*. [122] modeled subsystem cause-effect relationships as discriminative Boolean networks, composing modular rules to derive minimal fault-path explanations for fault diagnosis.

D. Group 3: Descriptive-Retrieval Approaches

The third significant aggregative design trend of DSR-AI approaches is the incorporation of knowledge with descriptive function under retrieval mechanism, *i.e*., Descriptive-Retrieval approaches. Generally, approaches following this pattern tend

to be particularly relevant to tasks characterized by *standardization* and *translation* requirements, including compilation, structuralization, and transcription processes.

a) Pure Descriptive-Retrieval approaches: Knowledge with descriptive function and retrieval mechanisms exhibit a natural conceptual alignment: knowledge with descriptive function serves as "the heuristic for the search space", specifically functioning as the organizational logic for spaces consisting of observable solutions or observable materials for solution construction; meanwhile, retrieval mechanisms navigate that space by systematically extracting relevant information to guide the exploration process. To demonstrate how specific applications can abstract to this function-mechanism combination, consider Dagdelen *et al*.'s [128] approach to scientific literature processing. In order to convert unstructured research papers into machine-readable databases, they observed that scientific texts contain structural patterns that could be systematically organized. Based on this insight, they formulated descriptive knowledge as schemas defining scientific entities and their relations, serving as an organizational framework for information extraction, and implemented retrieval through schema-guided named-entity slot filling with LLMs to systematically extract structured information from unstructured text.

Descriptive-Retrieval approaches span diverse domains, often focusing on standardization and translation tasks that demand structured knowledge and efficient retrieval. In robotic chemistry, Rauschen *et al*. [126] standardized lab protocols in a chemical DSL capturing experimental semantics, retrieving robot-specific action plans through capability matching. In time-critical cyber-physical systems, Hu *et al*. [125] made timing semantics explicit in a descriptive DSL, retrieving candidate specifications via grammar-guided exploration to improve synthesis accuracy. For medical QA, Yang *et al*. [129] structured clinical knowledge with descriptive indices and used a dual retriever–ranker Retrieval-Augmented Generation (RAG) pipeline to surface and prioritize guideline passages. In code intelligence, Gu *et al*. [90] leveraged DSL syntax and API idioms as descriptive layers, retrieving structured exemplars to reduce hallucinations and improve accuracy. Similarly, Zirak and Hemmati [127] enhanced automated program repair by retrieving project-aware exemplars and patch templates guided by descriptive grammars and bug–fix taxonomies.

b) Descriptive-Retrieval with Discriminative-Optimization approaches: Several hybrid approaches incorporate additional knowledge functions and utilization mechanisms in parallel with Descriptive-Retrieval frameworks. One prominent variant involves Descriptive-Retrieval & Discriminative-Optimization approaches. This hybrid methodology combines the search efficiency afforded by Descriptive-Retrieval mechanisms with the constraint enforcement capabilities offered by Discriminative-Optimization processes. Such integrated mechanisms naturally address the computational requirements of fixing and manipulating tasks, wherein modifications are systematically searched under the guidance of descriptive knowledge, while the quality and validity of resulting changes are evaluated through discriminative knowledge assessment. For example, Mehr *et al*. [150] addressed chemical automation by translating free-text synthesis procedures into executable robotic protocols. They encoded a machine-executable DSL as descriptive knowledge for standardized procedure retrieval and applied discriminative knowledge for syntax, safety, and capability validation, using closed-loop optimization to refine protocols through execution feedback.

Building on this paradigm, subsequent works have extended the Descriptive-Retrieval & Discriminative-Optimization framework to other scientific and computational domains. Tang *et al*. [153] introduced BioCoder, combining descriptive interfaces with discriminative unit-test harnesses, retrieving exemplars for in-context learning and refining LLMs through test-time optimization. In mathematical program synthesis, Romera-Paredes *et al*. [152] framed a compact DSL with discriminative proof oracles, using population-based retrieval and evolutionary optimization. For quantum program repair, Li *et al*. [154] proposed to use descriptive gate libraries and discriminative unitary checks, retrieving buggy circuit patterns and optimizing unitary operations to restore correctness.

c) Descriptive-Retrieval with Descriptive-Optimization approaches: Another notable approach encompasses joint mechanisms of retrieval and optimization (Descriptive-Retrieval & Descriptive-Optimization, both utilizing descriptive function). This approach extends one step beyond Descriptive-Retrieval approaches by leveraging descriptive knowledge not only to regulate and organize the current solution space, but also to directly facilitate the identification of more optimal solutions through guided optimization processes. Liu *et al*. [149] exemplified this approach in structured output generation by organizing format requirements (JSON, XML, Markdown) as descriptive knowledge covering schema, semantics, and style. They applied retrieval to match prompts with constraint templates and optimization to iteratively check and repair outputs for format compliance. For software code editing, Yen *et al*. [105] introduced a multi-level code-block DSR modeling APIs, functions, and statements as a descriptive scaffold for programs. Retrieval locates and reuses relevant code blocks, while optimization performs block-wise repair guided by error description from the compiler, enabling faster and controllable code editing than monolithic prompting.

d) Descriptive-Retrieval with Descriptive-Composition approaches: Additionally, joint mechanisms of retrieval and composition represent another significant advancement beyond Descriptive-Retrieval approaches. This approach utilizes descriptive knowledge not only to guide searching within the solution space but also to systematically combine multiple retrieved solutions into coherent composite results, proving particularly effective when solutions are selected from specialized libraries and subsequently organized for practical application. For example, Chen *et al*. [156] tackled chart visualization by organizing charts through structural and contextual properties encoded as descriptive knowledge. They used retrieval to find semantically similar visualizations and composition to combine retrieved specifications into new, context-aware chart recommendations aligned with human cognition.

In Descriptive-Retrieval with Descriptive-Composition settings, laboratory automation exemplifies how a DSR standardizes observables to power search and assembly. Steiner *et*

al. [18] encoded organic synthesis in an executable chemical DSL, retrieving reagents and operations from modular catalogs to compose device-agnostic robotic workflows. Rohrbach *et al*. [151] extended this by curating a validated chemical reaction database enabling retrieval and composition of verified synthesis routes. Beyond chemistry, Shaikhha *et al*. [155] modeled incremental linear algebra as a descriptive calculus, retrieving and chaining pattern templates to synthesize provably correct, efficient analytic programs.

e) Descriptive-Retrieval with Descriptive-Planning approaches: Furthermore, joint mechanisms of retrieval and planning constitute a related but distinct approach that shares similarities with composition-based methods introduced in the preceding paragraph. The primary distinction lies in the sequential organization of results as coherent trajectories rather than static combinations, thereby enabling dynamic solution pathways that unfold over time or across multiple decision points. For example, Carmona and Yu [158] tackled dual-arm robot planning by organizing bimanual tasks through structured task-plan annotations defining ordered primitives, objects, and coordination patterns as descriptive knowledge. They applied retrieval to locate relevant plan components and planning to chain these primitives into feasible, human-like action sequences. Complementarily, O'Donoghue *et al*. [157] automated biology experimentation by encoding wet-lab Standard Operating Procedures (SOPs) in a pseudocode DSL capturing actions, reagents, and control flow as descriptive knowledge. Retrieval surfaced reference snippets and grounded natural-language inputs, while planning reconstructed executable step sequences, enabling apples-to-apples comparison of LLM-based scientific planning across labs.

E. Local Practice Trends

In addition to the three groups exhibiting significant design trends, several local practice trends emerge that, while not demonstrating the same breadth of application scenarios or depth of integration with other knowledge functions and utilization mechanisms, represent insightful converged selections of techniques from practitioners across diverse backgrounds.

a) Descriptive-Planning with explanation mechanism: Descriptive-Planning & Descriptive-Explanation approaches provide a structural space, typically through syntax templates or regular expressions, to constrain the resulting outputs while simultaneously planning with these structures to synthesize solutions according to given observations. This approach is commonly accepted in AI-enhanced agent or robot planning tasks, which demand high robustness. In these applications, AI-generated planning trajectories are regulated by primitive action and state transition grammars during the agent's or robot's perception and interaction with the physical environment. For example, Yang *et al*. [163] addressed software security by structuring binary analysis around descriptive knowledge of instruction semantics, calling conventions, and control-/data-flow patterns. They used planning to reason over structured code representations for vulnerability detection and explanation to provide interpretable feature attributions to support analysts' decision-making.

Descriptive-Planning–Explanation approaches span diverse domains. In the realm of CAD, Li *et al*. [161] encoded construction sequences as a parametric DSL, planning valid command chains and explaining them by aligning strokes, constraints, and inferred operations. For aircraft maintenance, Xie *et al*. [102] organized manuals and fault trees into a troubleshooting graph, using retrieval-augmented planning and citation-grounded explanations for transparent diagnostics. In biologically inspired computing, Davidson *et al*. [165] defined goals as reward-generating programs, where plans were execution traces and explanations arose from human-readable clauses. In table-based fact verification, Wu *et al*. [166] linked claims to evidence graphs over table cells, planning multi-hop traversals and explaining results through interpretable subgraphs. For clinical analysis, Wu *et al*. [162] formalized causal protocols to plan compliant pipelines and explain results through causal justifications and audit trails. In materials discovery, Kikkawa and Ohno [164] planned experimental allocations in a descriptive design space, with explanations derived from uncertainty diagnostics and rationale tracking. Collectively, these works illustrated how descriptive DSRs made planning tractable and explanations faithful.

b) Generative-Modularization with Generative-Retrieval/Planning approaches: Some approaches integrate Generative-Modularization with Generative-Retrieval/Planning, using the generative function to modularize the system into reusable component libraries. Subsequently, these component libraries provide structural metric spaces for retrieval operations, such as galleries of templates, or serve as primitive sets to support planning activities. This approach is particularly well-suited for computer graphics applications. For example, Zhang *et al*. [88] addressed 3D scene understanding by representing scenes through interoperable modalities—programs, words, and embeddings—captured as generative knowledge in a tri-DSR. They modularized these entities into executable components, and applied planning to understand and edit scenes while maintaining structural consistency.

Building on the Generative-Modularization and Generative-Retrieval/Planning paradigm, Raistrick *et al*. [159] procedurally generated open-world 3D environments by modularizing physical and ecological processes into reusable modules, retrieving assets to match scene goals, and using the planning mechanism to sequence terrain synthesis, asset placement, and rendering for physics-law-consistent worlds. Extending to indoor scenes, Raistrick *et al*. [160] modeled indoor semantics, including room types, furniture, and lighting, as generative knowledge modularized by spatial structure. They retrieved assets via co-occurrence priors and planned layouts under physical constraints to produce photorealistic indoor scenes. Beyond graphics, Ge *et al*. [103] treated tool-use as generative knowledge of skills and task graphs, modularized domain tools into reusable libraries, retrieved tools by semantic alignment, and planned multi-step toolchains with LLM feedback to generalize expert procedures across domains.

c) Pure Generative-Explanation approaches: Several approaches incorporate knowledge with generative function and explanation mechanisms. Knowledge with generative

function serves as a simulation program of either a physical system or a knowledge system. This approach represents an inverse process to Discriminative-Explanation, where only observables are considered. Such methodology echoes the classical *learning-by-synthesis* approach originally proposed by David Marr [205]. Rather than directly utilizing the shortcuts provided by second principles to explain observations, first principles are employed to simulate the observations to gain explanations. Although this may appear to be a detour, the advantages lie in the enhanced reliability of the explanations, making such approaches particularly appropriate for reliability-demanding applications, such as scientific discovery. For instance, Ellis *et al*. [106] addressed linguistic theory building by modeling phonological and morphological patterns as generative knowledge through probabilistic programs encoding rule systems and latent structures. They applied explanation via Bayesian inference to identify compositional hypotheses that best reproduce observed linguistic data, producing interpretable, falsifiable theories.

Building on Generative-Explanation approaches, Williams [169] modeled scene understanding with structured generative models encoding part relations and image formation, using Bayesian inversion to infer scene hypotheses that explain observed pixels. Qi *et al*. [167] applied this paradigm to temporal perception, modeling human activities as grammar-driven generative processes and explaining videos through Earley parsing of sub-event structures for improved recognition and prediction. In human–robot interaction, Edmonds *et al*. [100] modeled robot task dynamics as behavioral generators and explained robot actions via contrastive hypotheses that enhanced user understanding and trust. For graphics authoring, Belouadi *et al*. [168] treated TikZ as a generative DSL, explaining figures by synthesizing executable programs that reproduce input renderings, yielding editable, structure-preserving vector outputs.

d) Discriminative-Retrieval with Discriminative-Planning approaches: There are Discriminative-Retrieval & Discriminative-Planning approaches that use knowledge with discriminative function as a lightweight perception module for complex environments, where the entire system, such as the physical environment, proves too complicated to simulate according to the required features. In such contexts, second principles are more appropriate than first principles. Therefore, selecting primitives from a large set of predicates and using these primitives for planning constitutes a practical choice for applications, where agents and robots are required to interact with dynamic environments in real time to solve tasks along their trajectories. As an example, Zhong *et al*. [101] addressed retrosynthesis prediction by modeling chemical reactions through discriminative knowledge of edit legality and substructure consistency. They used retrieval to select valid bond-editing actions and planning to assemble multi-step synthesis routes, achieving higher accuracy and chemical consistency than template-based baseline methods. Segler *et al*. [170] extended this idea by modeling retrosynthesis as a search guided by discriminative chemical priors, retrieving reaction templates and planning with MCTS to generate efficient synthesis routes.

Extending to robot assembly, Pane *et al*. [172] applied the discriminative-retrieval knowledge mechanism by encoding a CAD-aligned ontology of parts, tolerances, and skills. Relevant skills were retrieved via ontology reasoning and planned into executable sequences that respected physical constraints. Lioutikov *et al*. [171] learned an attribute grammar where predicates over robot movement primitives guided retrieval and grammar-constrained planning, producing safe, generalizable programs from examples. Coruhlu *et al*. [173] improved robot runtime robustness under partial observability by using discriminative monitors to retrieve key predicates from noisy data and plan recoveries with human-readable explanations. Beyond robotics, Yamagata *et al*. [174] built a discriminative ontology of cellular senescence, retrieving mechanism triples to plan hypothesis-driven biological studies. In materials discovery, Adams *et al*. [95] embedded expert constraints as discriminative knowledge in a Bayesian loop, retrieving informative experiments and planning next steps for faster, transparent phase-map convergence.

e) Pure Discriminative-Retrieval approaches: Some approaches integrate the discriminative function with the retrieval mechanism. The rationale of these approaches parallels that introduced in the preceding paragraph, with the distinction that the planning phase is supplanted by other AI approaches, such as LLMs. In this approach, the retrieval-only mechanism of knowledge with discriminative function serves as a reliable primitive set with predicates indicating possible environment states, which may originate from large corpora. To illustrate this abstraction, consider Shu *et al*. [89], who improved legal analytics tasks including Similar Case Retrieval, Precedent Recommendation, and Judgment Prediction. They observed that legal experts relied on distinctive patterns to differentiate between precedent and similar cases, as well as domain-specific legal syntax and terminology, and applied retrieval over large U.S. case corpora to reduce task confusion and improve zero- and few-shot performance.

Extending discriminative-retrieval to software log analysis, Qi *et al*. [176] tackled log QA by defining five discriminative reasoning operators that identified relevant records. A log retriever narrowed candidate logs before reasoning applied numeric and logical operations, improving accuracy and efficiency. In the realm of open-world agents, Wang *et al*. [175] used multimodal memory as discriminative knowledge, retrieving past visual and instruction traces to guide planning. This memory-augmented retrieval stabilized the decision-making process and enabled continual self-improvement across over 200 Minecraft tasks.

f) Pure Descriptive-Composition approaches: Descriptive-Composition approaches incorporate descriptive knowledge, typically captured either by a domain-specific library of a GPL or a DSL, to describe the environment that interoperates with the resulting solutions from other AI approaches, such as LLM-based code generators. Here, the approach serves as a translator from the current result to machine-executable programs. For example, considering Hou *et al*.'s [108] work to address the challenge of automatically generating GIS code from natural-language queries in geospatial analytics. They observed that geospatial

programming depends on domain-specific libraries, DSL syntax patterns, and map semantics to produce executable code. Based on this insight, they formulated descriptive knowledge as GIS libraries and map semantics as structured descriptors of the geospatial code space, and composed relevant GIS entities, natural-language descriptions, and code snippets to translate user intents into machine-executable geospatial programs while ensuring domain validity.

Building on the Descriptive-Composition paradigm, representative works span law, science, and engineering. In law, Soares *et al*. [179] developed a legal DSL that encoded document structures and composed statutes via grammar rules for coherent, machine-verifiable texts. Colombo *et al*. [185] extended this by codifying legal sources and instruction formats as descriptors, composing pretraining, fine-tuning, and preference optimization to enhance controllable legal LLM drafting. In science, Sun *et al*. [178] used molecular encodings as descriptors to compose feature selection and model ensembling for efficient organic photovoltaic material discovery, while Zheng *et al*. [186] combined molecular text representations and task prompts for improved molecular property prediction. Jumper *et al*. [96] composed sequence and structure descriptors through Evoformer modules for accurate protein folding, later extended by Roy and Al-Hashimi [184] to multi-molecule interactions. Greener *et al*. [177] used amino-acid and binding-site descriptions to compose latent decoding for novel metalloprotein design. In engineering, Willis *et al*. [180] composed neural program synthesis with human design traces in a descriptive CAD space for inverse design. Zhu *et al*. [183] composed morphological and learning descriptors to evolve modular robotic systems. Boiko *et al*. [181] described chemical languages and lab APIs to compose LLM-guided experimental workflows. Slack *et al*. [182] used feature and intent descriptions to compose goal-aligned model explanations.

g) Divide-and-Conquer approaches: The final practice trend generally encapsulates an interesting manifestation of the *divide-and-conquer* paradigm—approaches that simultaneously integrate decomposition/modularization and composition/planning mechanisms. The commonality among these DSR-AI approaches lies in the methodology that first obtains atomic components or reusable libraries through decomposition or modularization, serving as system primitives, and subsequently combines these primitives into solutions through composition or planning. This methodology finds particularly broad application in engineering-relevant domains, echoing fundamental systems engineering theory [206]. Across visualization, programming, robotics, and engineering, numerous works demonstrate the divide-and-conquer paradigm through modular decomposition and compositional synthesis.

In the realm of visual design, Jones *et al*. [199] treated visual artifacts as executable programs, decomposing edits and composing them toward target visuals with minimal supervision. Lyi *et al*. [191] developed Gosling, a modular genomics visualization grammar supporting reusable, constraint-aware designs, while Guo *et al*. [200] formalized a visual grammar for model-checking results to ensure correctness and interpretability. Extending to 3D, Jones *et al*. [93] built ShapeAssembly, which decomposed meshes into part-level macros and composed structured CAD hierarchies for editing.

Turning to software and programming, Coblenz *et al*. [99] structured language design via a task-focused DSL, decomposing workflows into design decisions. Meyers *et al*. [187] and Calinescu *et al*. [98] modularized temporal-verification models for scalable analysis. Yan *et al*. [201] integrated control- and data-dependencies into graph-aware reasoning for better code understanding. Korosteleva and Sorkine-Hornung [198] introduced GarmentCode to decompose garment geometry and compose executable sewing steps, while Gui *et al*. [203] developed UI-Copilot to decompose natural-language tasks and compose UI code with LLM-guided planning.

In robotics, Ritschel *et al*. [190] decomposed two-arm capabilities into primitives for block-based programming. Mao *et al*. [192] and Mon-Williams *et al*. [204] composed modular skill sequences for long-horizon control, while Chen *et al*. [195] decomposed task specifications into subgoals to synthesize improved robot programs.

Beyond these, several engineering works exemplify the divide-and-conquer paradigm through domain-specific decomposition and compositional synthesis. Carcangiu *et al*. [188] modularized gesture primitives and composed HMM-based recognition pipelines with explainable specifications, while Cata no [193] synthesized cyber-resilience programs by composing architectural tactics under network constraints. In data-intensive domains, Berger *et al*. [189] decomposed RNA-seq signals for constrained haplotype assembly, and Wan *et al*. [194] built CompressDB, composing rule-based compression and manipulation modules for high-performance querying without decompression. For engineering design, Wang *et al*. [196] created Vis4Mesh to visually decompose design objectives for interactive exploration, while Chen *et al*. [197] composed multi-criteria optimization for consistent BIM decisions. Finally, Oakes *et al*. [202] proposed modular ML workflows that decompose data, modeling, and evaluation into reproducible, cross-domain pipelines.

F. Lessons Learned from the Analysis

The comprehensive analysis of the three groupings and local practice trends yields valuable lessons for practitioners developing cutting-edge domain applications using DSR-AI approaches. Overall, the analysis results support the insights underlying the framework for disentangling domain context and methodology: DSR-AI approaches tailored for different applications can converge on knowledge function and utilization mechanism, while those within the same domain application can diverge on knowledge function and utilization mechanism due to differences in specific requirements.

Based on this analysis, we can summarize requirement-centric practices as described in Table II. The first column provides abstract requirement descriptions that discard specific domain context and focus solely on practitioner requirements, addressing the question: *"What task do you require the DSR-AI approach to perform?"* This column serves as the table index, clustered according to the groups based on function-mechanism pairs (see also the fourth column). The "Input" and "Output" columns help practitioners position their requirements more precisely. The following two columns, "role of

TABLE I
REPRESENTATIVE CASES ANALYZED AND CRITICIZED THROUGH THE TWO-DIMENSIONAL FRAMEWORK. THE TERMS IN THE COLUMN FUNCTION-MECHANISM ARE ABBREVIATED DUE TO SPACE LIMIT.

Func.-Mech.	Application Domain	Input	Output	Role of DSR	Role of AI
Gen.-Comp.	Viral evolution mechanism simulation [16]	Viral sequence data, mutation context.	Predicted viral escape variants.	Probabilistic grammar instantiating biological rules.	Grammar production rule applying; Sequence Composition.
Gen.-Comp.; Gen.-Mod./Decomp.	Novel music piece composition [138]	Musical pieces, composition tasks.	Theory-grounded musical compositions.	Music-theoretic knowledge set (pitch, duration, chord, form); Hierarchical score structures.	Theory-grounded musical primitives decomposing; Musical notes composition.
Gen.-Comp.; Dis.-Opt.	Automated robot structure design [130]	Design objectives, physical constraints.	Optimized robot morphologies.	Grammar for robot structures; Physics validator.	Robot design composition; Physical plausibility constraint optimization.
Gen.-Comp.; Dis.-Mod.	Robot task planning in automatic kitchen [139]	Natural-language cooking recipes.	Verifiable, reusable robot plans.	Conditioned atomic action set; Formal semantics and runtime executability checking.	Logic language specification composition; Queryable action schema modularization.
Dis.-Exp.	Human behavior understanding [117]	Video sequences, action queries.	Action detections with explanations.	Action taxonomy set; human-part states verification.	Graph-based human-part detection and explanation; Part relationship analysis.
Dis.-Exp.; Dis.-Opt.	Motion graphics animation [17]	Motion trajectories, physics context.	Physics-compliant animations.	Physical kinematics and dynamics logic predicates.	Physics violation explanation; Iterative refinement optimization.
Dis.-Exp.; Dis.-Mod./Decomp.	Human body posture parsing [143]	Human body images.	Body part segmentations.	Part-whole and contour-coherency grammar constraint set.	Structural reasoning for decision explanation; Signal decomposition into smooth components.
Dis.-Exp.; Dis.-Comp.	Industrial assembly quality control [120]	Product assemblies.	Logical anomalies.	Bill-of-materials logic; Assembly dependency constraint; logic code execution.	Missing or excessive component flagging.
Des.-Retr.	Scientific literature mining [128]	Research papers about solid-state materials.	Entity list with relationships.	Scientific entity and relation schema.	Schema-guided named entity slot filling; Structured information extraction.
Des.-Retr.; Dis.-Opt.	Automated chemical synthesis [150]	Literature synthesis procedures.	Executable laboratory protocols.	Synthesis procedure constraint; Execution verification.	Retrieval-based Protocol assembling; Closed-loop execution optimization.
Des.-Retr.; Des.-Opt.	Regulation of GenAI-user interaction [149]	User prompts, format requirements.	Constraint-compliant outputs.	Schema, semantics, style output constraint.	Constraint templates retrieving; Iterative output repairing.
Des.-Retr.; Des.-Comp.	Data visualization composition [156]	Visualization prompt and data.	Chart recommendations.	Chart structure constraint (marks, encodings, axes); Context data description.	Visualization retrieving; Specification composition.
Des.-Retr.; Des.-Plan.	Robot bimanual manipulation [158]	Video/mocap instances, task requirements.	Feasible bimanual action sequences.	Task-plan annotation framework; Orimitives, objects, and bimanual coordination tags.	Motion-based plan retrieval; Primitive chaining; Human-like sequences synthesizing.
Des.-Plan.; Des.-Exp.	Software security analysis [163]	Binary code, vulnerability patterns.	Vulnerability discoveries with evidence.	Instruction semantics and calling conventions; Control and data-flow patterns.	Structured code representation understanding; Similar reasoning planning; Rule-based interpretable feature explanation.
Gen.-Mod.; Gen-Retr./Plan.	3D scene understanding and generation [88]	Scene descriptions and images.	Editable scene programs.	Programs, words, and embeddings encoding; Shared scene factoring.	Executable operators and embeddings modularization; Ordered generation; Editing steps planning.
Gen.-Exp.	Linguistic theory analysis and discovery [106]	Human language corpora.	Falsifiable linguistic hypotheses.	Phonology and morphology constraints.	Theory programs posterior inferencing; Observation explanation; Interpretable hypotheses synthesizing.
Dis.-Retr.; Dis.-Plan.	Retrosynthesis pathway planning [101]	Target molecules, reaction constraints.	Multi-step synthetic routes.	Edit legality verification; Product-reactant consistency verification.	Bond-editing actions retrieving and ranking, Edit sequences planning.
Dis.-Retr.	Legal document analytics [89]	Legal queries, case databases.	Similar cases and precedents.	Legal domain syntax set; Task-specific legal distinction.	Structured legal case analyzing; Case corpora retrieving.
Des.-Comp.	Geospatial information analytics [108]	Geospatial analysis tasks.	Correct geo-code and summaries.	GIS map semantics library; DSL syntax and semantic library.	GIS entity retrieving, NL description synthesizing; Geospatial solution code synthesizing.

DSR" and "role of AI", provide design choices for practitioners selecting approaches to integrate DSR with AI and incorporate domain knowledge according to their requirements. The final column lists corresponding representative works. The decomposition/modularization-composition/planning pipeline, one of the local practice trends, is not included in the table because it represents a more abstract design pattern that cannot be simplified into specific requirements.

Diving into the design trends discovered through our analysis, we find that design choices based on requirements correlate with the function-mechanism configuration of DSR-AI approaches. Here we provide an intuitive understanding of the mapping between requirements and approaches to offer guidance accessible to practitioners without solid computer science and AI expertise. The readers are encouraged to comparatively read Table I and Table II to understand how the principles are abstracted from the specific applications.

The Generative-Composition methodology addresses the requirement of creating larger entities from smaller components. Incorporating Generative-Modularization/Decomposition addresses the source of these smaller components. Incorporating Discriminative-Optimization facilitates assessment from alternative perspectives, and resolves the difficulty of obtaining smaller components from complex larger entities. Discriminative-Explanation addresses the requirement of explaining new observations based on existing knowledge. Incorporating Discriminative-Optimization leverages explanations to search for better solutions. Incorporating Discriminative-Modularization/Decomposition addresses the origin of existing knowledge. Incorporating Discriminative-Composition helps when there is an abundance of existing knowledge. Descriptive-Retrieval addresses the requirement of selecting items from a larger set organized by specific logic. Incorporating Discriminative-Optimization helps refine solutions by understanding the target user. Incorporating Descriptive-Optimization further uses organizational logic to regulate solutions and enhance desirability. Incorporating Descriptive-Composition helps when the items to select are numerous. Incorporating Descriptive-Planning facilitates the additional requirement of selecting items with sequential decision-making.

Local practice trends also reveal more specific and insightful patterns: Descriptive-Planning & Descriptive-Explanation regulates agent behavior during interaction with the environment. Generative-Modularization & Generative-

TABLE II
SUMMARIZATION OF LESSONS LEARNED FROM THE ANALYSIS FOR CRAFTING DSR-AI APPROACHES. THE TERMS IN THE COLUMN FUNCTION-MECHANISM ARE ABBREVIATED DUE TO SPACE LIMIT. "OTH." IS THE ABBREVIATION OF "OTHERS IN THE PIPELINE".

Requirement	Input	Output	Func.-Mech.	Role of DSR	Role of AI	Representatives
Create structures or systems from given atomic components.	Informal description of creation intention; Given set of atomic components.	Target structures or systems aligned with creation intention.	Gen.-Comp.	Primitive set; Partially production system.	Partially production system; Others in the pipeline.	[109], [94], [16], [110], [111], [112], [113], [114], [115]
Create new structures or systems from atomic components analyzed from the original ones.	Informal description of creation intention; Original structures or systems.	New structures or systems aligned with creation intention.	Gen.-Comp.; Gen.-Mod./Decomp.	Primitive set; Partially inverse analysis; Partially production system.	Partially perception-related inverse analysis; Partially production system; Others in the pipeline.	[136], [91], [137], [138], [97], [88]
Create structures or systems from given atomic components while fixing the result by assessments.	Informal description of creation intention; Given set of atomic components; Given assessment metrics.	Target structures or systems aligned with creation intention and assessment metrics.	Gen.-Comp.; Dis.-Opt.	Primitive set; Partially production system; Partially assessment metrics.	Observation perception; Partially production system; Partially assessment metrics; Oth.	[130], [131], [132], [133], [134], [135]
Create shortcut pathways from atomic components simplified from the originally complicated systems.	Informal description of creation intention; Originally complicated systems.	New systems aligned with creation intention and the intrinsic dynamics of the original systems.	Gen.-Comp.; Dis.-Mod.	Primitive set; Partially production system; Metarules for simplification.	Partially production system; Perception modules for simplification; Others in the pipeline.	[139], [140]
Provide plausible explanations or guardrails for given observable results.	Given observable results; Given hypothesis or constraint space.	Target explanations or constrained outcomes aligned with observable results.	Dis.-Exp.	Rules spanning the hypothesis space; Partially explanation proposing.	Partially explanation proposing; Observation perception; Oth.	[107], [116], [117], [118], [119], [120], [121], [122], [123], [124]
Provide explanations for observable results of the evaluated subject, update the subject accordingly.	Given observable results; Given hypothesis or constraint space; Subject and its operational space.	Target explanations aligned with observable results; Update subject according to the explanations.	Dis.-Exp.; Dis.-Opt.	Rules spanning the hypothesis space; Partially explanation proposing; Partially operational space.	Partially explanation proposing; Partially operational space; Others in the pipeline.	[145], [146], [147], [148], [104], [17]
Provide explanations for observations where the hypothesis space is obtained by analyzing the subject.	Given observable results; Rules for analyzing the hypothesis space.	Elements spanning the hypothesis space; Target explanations aligned with observable results.	Dis.-Exp.; Dis.-Mod./Decomp.	Rules for analyzing the hypothesis space; Partially explanation proposing.	Partially explanation proposing; Observation perception; Others in the pipeline.	[141], [2], [142], [143], [92], [144], [17]
Provide explanations for observations where the hypothesis space is large with unions of explanations.	Given observable results; Given hypothesis space without tractable configuration.	Target union of explanations aligned with observable results.	Dis.-Exp.; Dis.-Comp.	Partially approximation of hypothesis space; Partially explanation proposing.	Partially approximation of hypothesis space; Partially explanation proposing; Observation perception; Oth.	[2], [120], [122]
Select a desirable solution from given solution space.	Given solution space; Given selection criteria.	Target solution aligned with selection criteria.	Des.-Retr.	Solution space structure; Partially selection criteria.	Partially selection criteria; Others in the pipeline.	[125], [126], [127], [128], [129], [90]
Select a solution from given solution space while fixing it by assessments.	Given solution space; Given selection criteria; Given assessment metrics.	Target solution aligned with selection criteria and assessment metrics.	Des.-Retr.; Dis.-Opt.	Solution space structure; Partially selection criteria and assessment metrics.	Observation perception; Partially selection criteria, assessment metrics; Oth.	[150], [151], [152], [153], [134], [154]
Regulating the solution space for selecting and fixing desirable solutions.	Given solution space; Given regulation criteria; Given fixing operations.	Updated target solution aligned with regulation criteria.	Des.-Retr.; Des.-Opt.	Solution space structure; Fixing operations; Regulation criteria.	Candidate solution proposing; Others in the pipeline.	[149], [105]
Select a desired set of solutions from given solution space and combine the set.	Given solution space as a reference library; Given selection criteria.	Target combined solution set aligned with selection criteria.	Des.-Retr.; Des.-Comp.	Solution library structure; Partially selection criteria and production system.	Partially selection criteria; Partially production system; Others in the pipeline.	[18], [150], [155], [151], [156]
Select solutions temporally from given solution space as desired plans.	Given solution space as a reference library; Given decision-making criteria.	Target trajectory of solutions aligned with decision-making criteria.	Des.-Retr.; Des.-Plan.	Solution library structure; Partially decision-making criteria, production system.	Partially decision-making criteria; Partially production system; Oth.	[157], [158]
Regulating agent/robot behavior and shaping resulting trajectories.	Given environment, action, and state spaces; Given decision-making criteria.	Target trajectory of agents/robots aligned with decision criteria.	Des.-Plan.; Des.-Exp.	Structural action and state spaces; Partially decision-making criteria.	Trajectory proposing; Partially decision-making criteria; Others in the pipeline.	[161], [162], [163], [164], [158], [165], [102], [166]
Selecting templates and primitives for procedural generation of creations.	Informal description of creation intention; Template gallery and primitive set.	Target outcomes of procedural generation aligned with creation intention.	Gen.-Mod.; Gen.-Retr./Plan.	Template gallery structure; Primitive set; Partially production system.	Partially production system; Others in the pipeline.	[159], [103], [160], [88]
Provide model- or theory-grounded explanations for given observable results.	Given observable results; Given primitive set within model or theory.	Target explanations aligned with observable results and grounded within theories.	Gen.-Exp.	Primitive set; Partially production system.	Partially production system; Observation perception; Others in the pipeline.	[100], [167], [106], [168], [169]
Planning within complicated environment with reliable guardrails.	Given environment and interaction knowledge; Given decision-making criteria.	Target trajectory of agents/robots aligned with decision criteria.	Dis.-Retr.; Dis.-Plan.	Predicates about interaction knowledge; Partially decision-making criteria.	Observation perception; Partially decision-making criteria; Oth.	[170], [171], [172], [173], [101], [174], [95]
Planning within complex environment with prior knowledge instances.	Given environment and knowledge instance; Given decision-making criteria.	Target trajectory of agents/robots aligned with decision criteria.	Dis.-Retr.	Predicates of prior knowledge instances regarding environment truth value.	Full pipeline of trajectory planning.	[175], [89], [176]
Translating given results to machine-executable programs.	Given results without standardized format.	Target programs executable by specific devices according to requirements.	Des.-Comp.	GPL or DSL according to requirements.	Previous result generation; Code generation; Others in the pipeline.	[177], [178], [179], [96], [180], [181], [182], [183], [184], [185], [186], [108]

Retrieval/Planning facilitates template selection and procedural generation in computer graphics. Generative-Explanation satisfies reliability-driven analysis demands. Both Discriminative-Retrieval & Discriminative-Planning and Discriminative-Retrieval facilitate agent planning within complex environments, though the latter requires more lightweight environmental knowledge than the former. Descriptive-Composition generally transforms natural outcomes into formal executable programs. The decomposition/modularization-composition/planning pipeline methodology can serve as a *gold standard* when practitioners have limited options and must attempt a solution, particularly when facing engineering challenges by simplifying the problem.

In summary, the key to making appropriate design choices for DSR-AI approaches lies in understanding the essence of the application task—looking beyond the superficial domain context of the target task to recognize that methodologies matter according to task requirements, inputs, and outputs. After determining the knowledge function and utilization mechanism for the DSR-AI approach, DSR implementation depends on specific contexts, with several summarized practices available for reference: (i) if established DSRs already exist in the

domain and are appropriate for the task with no adaptation or only slight modification, employing those DSRs is encouraged for simplicity; (ii) if appropriate DSRs do not exist, the practical choice is to craft minimal usable subsets that satisfy requirements—although this may require more iterations, it is more practical than directly designing a DSR with full capabilities; and (iii) while specific DSR constructs can be determined during development, the DSR implementation type should be determined in advance—if required features only involve syntactic formatting, data schemes suffice; if semantics are required, declarative models and notation systems should be considered; if knowledge should evolve systematically "like a machine", formal grammar should be used; and finally, if knowledge requires direct grounding to be effective, execution specifications should be employed. At this point, it becomes clear that when practitioners have to develop new DSRs, the effort becomes significantly challenging, and thereby, costly, raising a critical question that need to be answered in the remainder of this survey: *Are DSR-AI approaches sufficiently practical for broad usage?*

V. Challenges and Research Opportunities

This section discusses the challenges and research opportunities regarding the broader usage of DSR-AI approaches. To achieve effective coexistence with DSRs, the principal challenges encompass the substantial costs involved in manually designing, deploying, and communicating DSRs (Sec. V-A). These challenges highlight promising research thrusts concerning the automated design, deployment, and communication of DSRs (Sec. V-B), the scientific evaluation of such automated approaches (Sec. V-C), and the ecosystem-level implications of coexistence with DSRs (Sec. V-D).

A. On the Hardness of Coexistence with DSRs

Despite the advantages and emerging trend of computablizing human knowledge for AI through DSRs, the design, deployment, and communication of DSRs for specific requirements remain challenging for practitioners. This difficulty can be attributed to the inherently domain-specific nature of DSRs.

A DSR within a specific domain serves as a twin of human knowledge in that domain—a computable twin. Like two sides of the same coin, the DSR constructs that mirror knowledge and those that render knowledge computable are inextricably intertwined: for knowledge, computablization provides meaning, while for computablization, knowledge provides meaning. Consequently, developing a DSR requires a collaborative cohort comprising two distinct groups: one with domain expertise and another with information science background. For most domains, whose paradigms differ substantially from those of information sciences, mastering both the domain itself and information sciences proves impractical [68]. This necessitates deep collaboration between domain practitioners and information scientists, with each party extending beyond their current knowledge scope to understand the other's ideas and requirements by adopting their perspective [207]. Furthermore, the outcomes of such collaborative efforts typically apply only to the current domain of application and may not be reusable for other purposes [208]. These circumstances render DSR design labor-intensive, case-specific, and thereby costly.

Even when a DSR is ultimately developed after considerable effort, challenges persist. According to our findings regarding DSR-AI approaches, one primary method for instantiating the integration between DSRs and AI involves leveraging the code-generation capabilities of LLMs to synthesize and manipulate DSRs. However, unlike the primary targets of code generation [209], *i.e.*, synthesizing GPLs programs such as C++ and Python, for which ultra-large-scale training data are readily accessible—training data for synthesizing application-specific DSRs remain scarce because many target artifacts are newly designed and have limited usage histories.

Some beneficial attempts have been made to leverage prompt engineering to enable LLM-based code generators to synthesize DSL programs, yet the target DSLs are typically well-established with open-source projects developed within the corresponding DSL [132], exemplifying the scope for further research. In other technical pathways, deploying DSRs may require supporting infrastructure such as an interpreter for a data schema, a production system for a CFG, and a compiler for a DSL—these are also case-specific due to the diverse purposes of DSRs. Moreover, DSRs may require updating to align with the potentially rapid evolution of domain-specific knowledge, making DSR maintenance a continuous challenge. This situation also presents opportunities to modularize DSRs, preventing them from becoming "big balls of mud" while increasing the reusability of DSR constructs between different DSRs, particularly those for closely related applications. For example, DSRs for describing biological and chemical experiments share constructs regarding standard experimental practices [210]. Additionally, different sources of DSRs may need to function as an ensemble for shared objectives or within the same system, potentially requiring busline-like communication mechanisms that could have broader impacts, such as facilitating interdisciplinary cooperation.

In summary, these challenges underscore the difficulty of coexisting with DSRs in the AI era while simultaneously highlighting valuable potential research directions.

B. Automating the Coexistence with DSRs

Given the complexity inherent in designing, deploying, and communicating DSRs, an intuitive solution involves automating these processes to some extent, thereby reducing human effort in the workflow and making DSRs more accessible to a broader spectrum of domains. This represents a challenging yet valuable research direction: given the increasing significance of DSR, reducing the cost of coexistence with DSRs across all facets becomes imperative.

Understanding and reproducing the exact processes of DSR development presents significant challenges. For already-established DSRs, the "story behind the design", namely, the rationale underlying design decisions, is inevitably intricate, making it nearly impossible to precisely identify the knowledge employed in proposing a single construct. The domain-specific knowledge that guides design may not be readily accessible, as it can manifest in arbitrary modalities. These

TABLE III
SUMMARIZATION OF CURRENT ATTEMPTS ON AUTOMATING THE COEXISTENCE WITH DSRS. THE TERMS IN THE COLUMN FUNCTION-MECHANISM ARE ABBREVIATED DUE TO SPACE LIMIT. "COMM." IS THE ABBREVIATION OF "COMMUNICATION".

Objective	Implementation	Source of Domain Knowledge	Application Scenario	Func.-Mech.	Ref.
DSR design	Imperative DSL	Self-contained procedure documentation	Formalization of protocols for scientific experiments	Des.-Retr. & Des.-Opt.	[211]
DSR design	Declarative DSL	Examples sampled from commonsense knowledge base	Functionalization of human instructions for 3D modeling	Gen.-Mod. & Gen.-Comp.	[212]
DSR design	Imp. & Dec. DSL	Self-contained procedure documentation	Abstraction of atomic steps for experiment protocol planning	Dis.-Retr. & Dis.-Plan.	[213]
DSR design	Imp. & Dec. DSL	Procedure documentation, configurations, and records	Extraction of constraint relations for scheduling optimization	Dis.-Retr. & Dis.-Plan.	[214]
DSR design	Declarative rules	Raw data, summarized practices, and domain knowledge	Generation and constraint of data-driven narratives	Gen.-Comp. & Dis.-Opt.	[215]
DSR deploy	Imperative DSL	Self-contained procedure documentation	Compilation from NL protocols to machine-executable programs	Des.-Comp. & Dis.-Retr.	[216]
DSR deploy	Imperative DSL	Distribution of procedures executing on the hardware	Optimization of minimal hardware to satisfy scheduling need	Des.-Comp. & Dis.-Opt.	[217]
DSR comm.	Data schema	NL corpora of published papers	Interpretation on a cluster of papers for different perspectives	Dis.-Exp.	[218]
DSR comm.	First-order rules	Distribution of natural visual concepts across domains	Generalization of visual concepts through similarities and rules	Dis.-Decomp.	[219]

range from idealized self-contained textual documentation, to design diagrams, reverse engineering from existing products, perceptual outcomes, or even commonsense knowledge bases, among others. Meanwhile, both the knowledge utilized by domain specialists and information scientists can constitute tacit knowledge, such as empirical judgments and spontaneous insights, which cannot be precisely quantified. These remain open problems requiring further investigation.

Several early, promising attempts have been made to take the first step toward automating coexistence with DSRs. Shi *et al*. proposed the Automated Design of Domain-Specific Language (AutoDSL) problem [211], which represents the first explicit formalization of the "domain corpora in, DSL out" workflow for paradigms featuring an explicit two-stage pipeline for domain solutions. This approach transforms the purely data-driven end-to-end pipeline into two stages: the first involves data-driven AutoDSL, and the second incorporates the resulting DSL with AI methods such as LLMs. This framework is particularly suited for precision- and robustness-demanding applications, such as designing imperative DSLs to represent experimental procedures across various domains including biology, medicine, and ecology for subsequent execution in self-driving laboratories. AutoDSL aims to fully leverage the representational power of the most flexible and comprehensive instantiation of DSR, namely, the DSL, to create DSR-AI approaches with high adaptability according to domain-specific requirements.

Following the proposal of AutoDSL, subsequent works have expanded its supporting programming paradigms, utilization methodologies, and application scenarios. The purely imperative programming paradigm for representing procedures has been extended to encompass declarative programming paradigms for systematic representation of objects along with their respective properties and methods [212], with applications in CAD modeling. Concurrently, imperative programming with dual-viewed DSLs for self-interleaved procedural validation to enhance robust representation has been explored [213], [214], leveraging interactions between the same system viewed from different perspectives [218]. Regarding utilization methodologies for the resulting DSLs from AutoDSL, retrieval mechanisms have been extended to include modularization and planning, where atomic operations are automatically abstracted to encapsulate interfaces, and the resulting interfaces serve as primitives for planning [213], [214]. These approaches have been applied to novel experimental procedure design and constraint specification for job scheduling. The resulting DSLs from AutoDSL can also be relaxed from rigorous DSLs to DSRs to accompany GenAI approaches for content creation, supporting hybrid methodologies that ensemble generative and discriminative functions of knowledge by defining a generative set for creativity and a constraint set for plausibility [215], with applications in GenAI for data-driven storytelling.

The sources of human knowledge for AutoDSL have also been progressively expanded. They range from relatively clean and single-modal corpora such as experimental protocols [211], [213], to heterogeneous multi-modal corpora including product orders, route sheets, and historical production data [214], and to the explicit capture of tacit knowledge from big-data-encoded prior commonsense knowledge within DNNs and LLMs, which describes the natural distributions of attributes bound to objects in the physical world [219], [212]. Given such knowledge sources in flexible, relaxed forms, an intuitive development has been the exploration of "programming with natural language", where NL-based instructions can be translated into programs using AutoDSL-resulting DSLs that are syntactically plausible, semantically correct, and ready for machine execution [216]. Furthermore, the communication gaps between high-level NL-based instructions and low-level machine code can be mitigated through AutoDSL-resulting DSLs [212]. Taking this concept further, the systems that generate data for AutoDSL can themselves be inversely manipulated by the resulting DSLs. For example, the hardware device layout of a self-driving laboratory can be optimized by controlling DSL programs through dependency analysis to identify layouts that minimally satisfy the desired flexibility of resource scheduling [217].

In summary, numerous challenges remain on the path to fully realizing the automation of DSR design, deployment, and communication for large-scale real-world applications—current explorations represent merely the "tip of the iceberg". Please refer to Table III to gain insights on the interconnections and limitations of these current attempts. Since DSRs bridge the originally divided boundaries between tacit and explicit knowledge, humans and computers, perception and reasoning, concept and execution, among others, they open considerable opportunities for interdisciplinary research.

C. Evaluating the Coexistence with DSRs

Following initial attempts with AutoDSL and subsequent developments, a practical challenge emerges: how to evaluate the quality of the resulting DSR? This evaluation presents significant difficulties because DSR constitutes "the technique behind" the pipelines of DSR-AI approaches. Consequently, direct evaluation and benchmarking of DSR-AI approaches cannot precisely reveal the quality of the resulting DSR, as confounding factors, such as the methodology for AI utilization of DSR or performance differences between the employed AI models, can significantly affect the evaluation results. This remains an open question, and we propose two potential solution pathways: direct assessment and controlled indirect assessment.

Direct assessment involves isolating the resulting DSR to apply evaluation criteria, following methodologies from domain engineering [220]. This approach has been extensively employed in scenarios where techniques must be adapted to specific requirements, such as software engineering.

For quantitative assessment, we typically check the mapping from the element set in the reference model, *i.e.*, concepts and relations indicating the domain ontology or dynamics, which may be incorporated within domain corpora, to the construct set of the DSR, focusing on four criteria: soundness, lucidity, completeness, and laconicity [221]. A DSR that ideally reflects domain ontology or dynamics should ensure that: (i) all reference model elements are mapped to the construct set for soundness; (ii) every reference model element is mapped to exactly one construct for lucidity; (iii) the construct set contains no redundancy beyond the reference model elements for completeness; and (iv) every construct refers to exactly one reference model element for laconicity.

For qualitative assessment, we typically employ the design guidelines of DSL as test scale questions for assessing DSRs from a user-centric perspective [222]. These questions range from objective consistency for machine interpretation to subjective complexity for user understanding. Specifically, the DSR should maintain: (i) simplicity, *i.e.*, being easy to understand both syntactically and semantically; (ii) clarity, *i.e.*, eliminating unnecessary space for generalization; and (iii) conciseness, *i.e.*, avoiding redundant concepts and relations.

Although direct assessment incorporates both objective and human-centric evaluation criteria, it typically supports comparison rather than direct scoring, as groundtruth and baseline DSR may not always be accessible. The absolute values of the four criteria may become distorted due to computational details when mapping the reference model to DSR constructs, particularly when dealing with heterogeneous domain data. Consequently, the reliability of such criteria may be diminished—a DSR with low soundness may not necessarily perform poorly in practice. Therefore, calibration of direct assessment criteria requires further research to enable context-appropriate adjustments.

Controlled indirect assessment typically involves replacing the original DSR with alternative versions or representations (including baselines enacted by LLMs) while keeping all other pipeline module parameters fixed, thereby creating a controlled-experiment [223]. The input-output configurations of the alternatives to the original DSR must remain identical to those of the original DSR. Through this rigorous variable control protocol, the final pipeline outputs can be compared, and differences between outputs can be attributed, with high probability, to differences between the original DSR and its counterparts. This protocol requires fine-grained considerations for practical implementation, representing a valuable research direction.

D. Constructing a DSR-Friendly Ecosystem

Coexistence with DSRs presents challenges at both the technical and governance levels. From the macro perspective of information society, achieving such coexistence requires a systematic strategy for leveraging distributed human knowledge throughout society. The concept of *distributed* can be interpreted across multiple dimensions: spatially, temporally, and functionally distributed.

Along the spatial dimension, knowledge-intensive data required for creating DSRs are commonly distributed across disparate sources, presented in heterogeneous formats and modalities. These data may remain unaggregated due to business privacy concerns or technical barriers. For instance, different Original Equipment Manufacturers (OEMs) in the automotive industry may have accumulated distinct insights from empirical production challenges, yet they typically refrain from sharing such knowledge with competitors [224], despite potential industry-wide benefits. This phenomenon represents the informational manifestation of organizational silos [225]. Furthermore, robots from different manufacturers are typically equipped with proprietary, encapsulated Operating System (OS) interfaces designed to emphasize their functional differentiation. This heterogeneity necessitates algorithm-specific adaptations for deployment across different robotic platforms [226]. To foster a viable DSRs ecosystem within such industries, standardized DSR-based information exchange protocols and unified OS abstraction levels may be essential prerequisites.

The temporal dimension presents challenges primarily related to tacit knowledge, particularly critical in advanced manufacturing and medical practice, which remains closely bound to individual expertise. Such tacit knowledge is typically transmitted through apprenticeship (often semi-transparent), which are inherently prone to information loss and inefficiency. This creates a significant risk: when the recruitment rate of new personnel falls below the retirement rate of senior experts, knowledge transmission may experience critical discontinuities, necessitating future "institutional archaeology" [3]. To mitigate such scenarios, the standardization and explicitization of tacit knowledge becomes imperative. The ecological niche that DSRs can play in this evolution requires careful consideration: determining which knowledge can be standardized and explicitized versus which knowledge can only be captured and communicated through human expertise.

[3]This term was first introduced in a blog article posted by Rob Landley at https://landley.net/history/mirror/institutional_memory.html.

The functional dimension is characterized by the mismatch of potentially synergistic resources—resources that could achieve synergistic effects when combined but remain distributed due to various constraints. For example, in rare disease research, medical records may be fragmented across different medical departments worldwide. While these fragments could collectively provide valuable insights for rare disease treatment, individual fragments remain difficult to access [227]. However, identifying meaningful connections between these fragments represents a classic *unknown unknowns* problem, where proactive searching for clues resembles "finding a needle in a haystack". In such situations, capturing the meta-knowledge of *"What is critical for connecting the dots?"* becomes significant. This meta-knowledge may serve as a prerequisite for developing relevant DSRs to facilitate the discovery of meaningful connections.

In summary, DSRs do not constitute universal solutions for making knowledge computable by AI systems. The application context determines whether DSR-AI approaches are appropriate, whether transforming specific knowledge into DSRs is meaningful, and which type of DSR to implement. These technical and societal contexts form the foundational prerequisites for successful integration with DSRs. These ecosystem-level considerations necessitate further interdisciplinary research to fully clarify the technical directions regarding DSRs.

VI. Discussion and Conclusion

This survey examines a growing class of AI practices in which explicit domain-specific knowledge representation artifacts make specialized knowledge computable for AI systems. Using DSR and DSR-AI as umbrella terminology, we trace the historical development of the underlying artifact families and their integration with AI, and position our cross-domain analysis relative to prior surveys. Our aim is to connect previously fragmented domain-specific attempts at DSR-AI approaches and provide insights for future development. To achieve this objective, we disentangle domain context from DSR-AI methodology through a two-dimensional framework that examines both the function of human knowledge incorporated and the mechanisms by which AI systems utilize this knowledge. Using this framework, we conduct a comprehensive literature review to map the landscape of DSR-AI approaches developed to address domain-specific problems, analyze the design trends underlying these techniques, and synthesize practical principles derived from our analysis. Beyond surveying the diverse tasks addressed by DSR-AI approaches, we identify a critical challenge concerning the practicability of their widespread adoption: their case-specificity and high implementation costs. We propose that the solution may lie in automating the design, deployment, and communication of DSRs through intelligent methods. This survey aims to contribute to a holistic understanding of cutting-edge DSR-AI approaches for domain applications and to provide generalizable practical principles for more standardized, systematic, and therefore more affordable development of DSR-AI approaches. Ultimately, we seek to democratize AI advances for a broader spectrum of domains worldwide, thereby promoting enhanced human well-being.

A. On the Generality of Representations

A longstanding debate exists regarding whether computable knowledge representations should be tailored to specific domains and requirements, namely, DSRs, or designed with maximum generality to encompass a broad range of application contexts, functioning as a *once-and-for-all* solution. Both perspectives offer distinct advantages and disadvantages, giving rise to the generality-specificity dilemma [228].

DSRs benefit from the injection of fine-grained, domain-specific knowledge, requiring only the subset of representational constructs pertinent to the target domain rather than all conceivable representational constructs. This specificity simplifies the operations needed for AI utilization, reduces complexity for human comprehension, and alleviates computational resource demands, thereby enhancing the efficiency with which DSRs mirror domain contexts. However, the primary drawback lies in the manual crafting of DSRs, which necessitates extensive collaboration between domain experts and computer scientists. This approach is inherently case-specific, labor-intensive, and consequently, costly. Thus, DSR embraces specificity at the expense of generality.

Conversely, consider the prospect of a general knowledge representation offering a once-and-for-all solution, constituting a *one-size-fits-all* representation incorporating all possible constructs for all possible knowledge domains. While such a general representation could theoretically be developed given sufficient investment, this approach parallels the paradox illustrated in Jorge Luis Borges' story *On Exactitude in Science*, wherein cartographers sought to create a perfect map. Such an endeavor would ultimately result in duplicating the world itself—the knowledge representation would need to match the world's complexity, rendering it too intricate for practical usage and defeating the original objective of computablization. The efficacy of task-solving resides in abstraction, selecting constructs relevant to specific tasks while discarding those unrelated (though they may prove relevant to other tasks). This specificity is inherently at odds with generality. Therefore, DSR appear to be a necessity rather than a choice. Rather than seeking an elusive generality, focusing on DSR adaptation and striving for the automation of this process may offer a pragmatic way to circumvent this dilemma.

B. A Prospect with Generalist and Specialist AI

Currently, there is lively discussion regarding whether Generalist AI—namely, computers or robots equipped with general skill sets to perform actions as diverse as those executed by humans—represents the ultimate solution in AI [229], [230]. Indeed, significant progress has been made in Generalist AI solutions for both virtual and embodied agents to assist humans across numerous tasks [11], [231], [232]. Despite this context, we maintain the position that specialized AI solutions designed for domain-specific tasks remain indispensable, with the potential for ensembles that combine both Generalist and Specialist AI approaches.

While a Generalist AI agent may inspire blue-sky thinking about Artificial General Intelligence (AGI), it may sacrifice performance and effectiveness in complex tasks that require

in-depth domain understanding compared to specialized AI solutions, which can systematically incorporate domain knowledge through the relevant DSR. From a macro perspective, given *limited resources*, a Pareto frontier exists among multiple design requirements. If we expect the resulting AI agent to excel at domain-specific tasks and even outperform humans, the design should maximize target and relevant requirements while minimizing irrelevant ones. Such optimization enables the agent to transition from being "capable of diverse tasks but excelling at none" to "excelling at a single, specialized task". This represents the natural, rational, and economic choice for engineering practice [233], where maximizing overall utility is the pivotal requirement.

Conversely, Generalist AI agents offer unique advantages when users specifically require human-like assistants, such as work companions or household robots. In these cases, humans prefer to interact with a single agent through a simple, unified interface, regardless of the complexity operating behind it. This parallels the distinction between high-level and low-level programming languages: the former provides users with an accessible interface for instruction delivery, while the latter executes programs on target hardware. In such application scenarios, Generalist and Specialist AI can function in ensemble through a hierarchical architecture, where a Generalist AI agent comprehends high-level user instructions and leverages its advantage in broad knowledge scope to decide and plan which Specialist AI to *call* for accomplishing specific tasks. The trend of DSR-AI approaches demonstrated in this survey provides space for envisioning how group optima can be achieved through collaboration among heterogeneous *machine experts*. This story of future may lack the excitement of a hero's journey, where a Generalist AI agent fights as a lone hero against problems and single-handedly saves the world—instead, victory may be attributed to cooperation among diverse Specialist AI solutions.

REFERENCES

[1] J. L. Russell, "Kepler's laws of planetary motion: 1609–1666," *The British journal for the history of science*, vol. 2, no. 1, pp. 1–24, 1964.

[2] S.-M. Udrescu and M. Tegmark, "Ai feynman: A physics-inspired method for symbolic regression," *Science Advances*, vol. 6, no. 16, p. eaay2631, 2020.

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems*, 2017.

[4] J. Wei, M. Bosma, V. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, and Q. V. Le, "Finetuned language models are zero-shot learners," in *International Conference on Learning Representations*, 2022.

[5] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, pp. 1–48, 2019.

[6] W. Haensch, T. Gokmen, and R. Puri, "The next generation of deep learning hardware: Analog computing," *Proceedings of the IEEE*, vol. 107, no. 1, pp. 108–122, 2018.

[7] J. Pei, L. Deng, S. Song, M. Zhao, Y. Zhang, S. Wu, G. Wang, Z. Zou, Z. Wu, W. He, *et al.*, "Towards artificial general intelligence with hybrid tianjic chip architecture," *Nature*, vol. 572, no. 7767, pp. 106–111, 2019.

[8] H. Wang, T. Fu, Y. Du, W. Gao, K. Huang, Z. Liu, P. Chandak, S. Liu, P. Van Katwyk, A. Deac, *et al.*, "Scientific discovery in the age of artificial intelligence," *Nature*, vol. 620, no. 7972, pp. 47–60, 2023.

[9] G.-Z. Yang, J. Bellingham, P. E. Dupont, P. Fischer, L. Floridi, R. Full, N. Jacobstein, V. Kumar, M. McNutt, R. Merrifield, *et al.*, "The grand challenges of science robotics," *Science Robotics*, vol. 3, no. 14, p. eaar7650, 2018.

[10] D. M. Lazer, A. Pentland, D. J. Watts, S. Aral, S. Athey, N. Contractor, D. Freelon, S. Gonzalez-Bailon, G. King, H. Margetts, *et al.*, "Computational social science: Obstacles and opportunities," *Science*, vol. 369, no. 6507, pp. 1060–1062, 2020.

[11] M. Moor, O. Banerjee, Z. S. H. Abad, H. M. Krumholz, J. Leskovec, E. J. Topol, and P. Rajpurkar, "Foundation models for generalist medical artificial intelligence," *Nature*, vol. 616, no. 7956, pp. 259–265, 2023.

[12] M. Janssen, P. Brous, E. Estevez, L. S. Barbosa, and T. Janowski, "Data governance: Organizing data for trustworthy artificial intelligence," *Government Information Quarterly*, vol. 37, no. 3, p. 101493, 2020.

[13] Z. Epstein, A. Hertzmann, I. of Human Creativity, M. Akten, H. Farid, J. Fjeld, M. R. Frank, M. Groh, L. Herman, N. Leach, *et al.*, "Art and the science of generative ai," *Science*, vol. 380, no. 6650, pp. 1110–1111, 2023.

[14] J. Kramer, "Is abstraction the key to computing?," *Communications of the ACM*, vol. 50, no. 4, pp. 36–42, 2007.

[15] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 2002.

[16] B. Hie, E. D. Zhong, B. Berger, and B. Bryson, "Learning the language of viral evolution and escape," *Science*, vol. 371, no. 6526, pp. 284–288, 2021.

[17] J. Ma and M. Agrawala, "Mover: Motion verification for motion graphics animations," *ACM Transactions on Graphics*, 2025.

[18] S. Steiner, J. Wolf, S. Glatzel, A. Andreou, J. M. Granda, G. Keenan, T. Hinkley, G. Aragon-Camarasa, P. J. Kitson, D. Angelone, *et al.*, "Organic synthesis in a modular robotic system driven by a chemical programming language," *Science*, vol. 363, no. 6423, p. eaav2211, 2019.

[19] J. F. Sowa, *Knowledge representation: logical, philosophical and computational foundations*. Brooks/Cole Publishing Co., 1999.

[20] D. Weininger, "Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules," *Journal of Chemical Information and Computer Sciences*, vol. 28, no. 1, pp. 31–36, 1988.

[21] B. Kuipers, "Qualitative simulation," *Artificial Intelligence*, vol. 29, no. 3, pp. 289–338, 1986.

[22] J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley Longman Publishing Co., Inc., 1996.

[23] J. Ellson, E. R. Gansner, E. Koutsofios, S. C. North, and G. Woodhull, "Graphviz and dynagraph—static and dynamic graph drawing tools," in *Graph drawing software*, pp. 127–148, Springer, 2004.

[24] D. B. Lenat, "Cyc: A large-scale investment in knowledge infrastructure," *Communications of the ACM*, vol. 38, no. 11, pp. 33–38, 1995.

[25] M. Fowler, *Domain-specific languages*. Pearson Education, 2010.

[26] M. Fox and D. Long, "Pddl2. 1: An extension to pddl for expressing temporal planning domains," *Journal of Artificial Intelligence Research*, vol. 20, pp. 61–124, 2003.

[27] R. K. Merton, *The sociology of science: Theoretical and empirical investigations*. University of Chicago Press, 1973.

[28] A. Newell and H. Simon, "The logic theory machine–a complex information processing system," *IRE Transactions on Information Theory*, vol. 2, no. 3, pp. 61–79, 1956.

[29] E. H. Shortliffe, "Computer based medical consultations: Mycin," *Elsevier*, 1976.

[30] T. R. Gruber, "Toward principles for the design of ontologies used for knowledge sharing?," *International Journal of Human-Computer Studies*, vol. 43, no. 5-6, pp. 907–928, 1995.

[31] B. Foote and J. Yoder, "Big ball of mud," *Pattern Languages of Program Design*, vol. 4, pp. 654–692, 1997.

[32] Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, 1998.

[33] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intelligent Systems and Their Applications*, vol. 13, no. 4, pp. 18–28, 1998.

[34] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, no. 1, pp. 81–106, 1986.

[35] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., 1988.

[36] D. L. McGuinness, F. Van Harmelen, *et al.*, "Owl web ontology language overview," *W3C Recommendation*, vol. 10, no. 10, p. 2004, 2004.

[37] B. Ø. Palsson, *Systems biology: properties of reconstructed networks*. Cambridge University Press, 2006.

[38] M. Richardson and P. Domingos, "Markov logic networks," *Machine Learning*, vol. 62, no. 1, pp. 107–136, 2006.

[39] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[40] Z.-H. Zhou, "Abductive learning: towards bridging machine learning and logical reasoning," *Science China. Information Sciences*, vol. 62, no. 7, p. 76101, 2019.

[41] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, vol. 378, pp. 686–707, 2019.

[42] I. Yildirim and L. Paul, "From task structures to world models: what do LLMs know?," *Trends in Cognitive Sciences*, 2024.

[43] S. Ouyang, J. M. Zhang, M. Harman, and M. Wang, "An empirical study of the non-determinism of chatgpt in code generation," *ACM Transactions on Software Engineering and Methodology*, 2024.

[44] E. F. Codd, "A relational model of data for large shared data banks," *Communications of the ACM*, vol. 13, no. 6, pp. 377–387, 1970.

[45] D. C. Fallside and P. Walmsley, "Xml schema part 0: primer second edition," *W3C Recommendation*, vol. 16, 2004.

[46] F. Pezoa, J. L. Reutter, F. Suarez, M. Ugarte, and D. Vrgoč, "Foundations of json schema," in *Proceedings of the International Conference on World Wide Web*, 2016.

[47] J. H. Davenport, Y. Siret, and É. Tournier, *Computer algebra: Systems and algorithms for algebraic computation*. Academic Press Ltd., 1988.

[48] H. Strubbe, "Presentation of the schoonschip system," *ACM SIGSAM Bulletin*, vol. 8, no. 3, pp. 55–60, 1974.

[49] A. Heck and W. Koepf, *Introduction to MAPLE*. Springer, 1993.

[50] S. Wolfram, *The mathematica book*. Wolfram Research, Inc., 2003.

[51] D. E. Knuth, "Semantics of context-free languages," *Mathematical Systems Theory*, vol. 2, no. 2, pp. 127–145, 1968.

[52] S. C. Johnson *et al.*, *Yacc: Yet another compiler-compiler*, vol. 32. Bell Laboratories Murray Hill, NJ, 1975.

[53] R. Corbett and R. Stallman, "Bison: Gnu parser generator," *Texinfo documentation, Free Software Foundation, Cambridge, Mass*, 1991.

[54] T. J. Parr and R. W. Quong, "Antlr: A predicated-ll (k) parser generator," *Software: Practice and Experience*, vol. 25, no. 7, pp. 789–810, 1995.

[55] A. V. Aho, M. S. Lam, and J. D. Ullman, *Compilers principles, techniques & tools*. Pearson Education, 2007.

[56] W. F. Clocksin and C. S. Mellish, *Programming in PROLOG*. Springer Science & Business Media, 2003.

[57] A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G. D. Melo, C. Gutierrez, S. Kirrane, J. E. L. Gayo, R. Navigli, S. Neumaier, *et al.*, "Knowledge graphs," *ACM Computing Surveys*, vol. 54, no. 4, pp. 1–37, 2021.

[58] K. Kukich, "Design of a knowledge-based report generator," in *Annual Meeting of the Association for Computational Linguistics*, 1983.

[59] C. J. Date, *A Guide to the SQL Standard*. Addison-Wesley Longman Publishing Co., Inc., 1989.

[60] S. E. Madnick and A. Moulton, "Script, an on-line manuscript processing system," *IEEE Transactions on Engineering Writing and Speech*, vol. 11, no. 2, pp. 92–100, 2007.

[61] J. Guo, "Usage and system management of bookmaker 6.02," in *Printing Technology*, China Institute of Printing Science and Technology, Beijing, 2000.

[62] L. Flavell, *Beginning blender: open source 3d modeling, animation, and game design*. Apress, 2011.

[63] B. Falck, D. Falck, and B. Collette, *Freecad [How-To]*. Packt Publishing Ltd, 2012.

[64] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, 2019.

[65] D. K. Chaturvedi, *Modeling and simulation of systems using MATLAB and Simulink*. CRC Press, 2017.

[66] P. Fritzson and V. Engelson, "Modelica—a unified object-oriented language for system modeling and simulation," in *European Conference on Object-Oriented Programming*, 1998.

[67] R. Lipsett, C. F. Schaefer, and C. Ussery, *VHDL: Hardware description and design*. Springer Science & Business Media, 1989.

[68] M. Mernik, J. Heering, and A. M. Sloane, "When and how to develop domain-specific languages," *ACM Computing Surveys*, vol. 37, no. 4, pp. 316–344, 2005.

[69] W.-Z. Dai, Q. Xu, Y. Yu, and Z.-H. Zhou, "Bridging machine learning and logical reasoning by abductive learning," in *Advances in Neural Information Processing Systems*, 2019.

[70] J. Mao, C. Gan, P. Kohli, J. B. Tenenbaum, and J. Wu, "The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision," in *International Conference on Learning Representations*, 2019.

[71] R. Feinman and B. M. Lake, "Learning task-general representations with generative neuro-symbolic modeling," in *International Conference on Learning Representations*, 2021.

[72] L. Von Rueden, S. Mayer, K. Beckh, B. Georgiev, S. Giesselbach, R. Heese, B. Kirsch, J. Pfrommer, A. Pick, R. Ramamurthy, *et al.*, "Informed machine learning–a taxonomy and survey of integrating prior knowledge into learning systems," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 1, pp. 614–633, 2021.

[73] T. Dash, S. Chitlangia, A. Ahuja, and A. Srinivasan, "A review of some techniques for inclusion of domain-knowledge into deep neural networks," *Scientific Reports*, vol. 12, no. 1, p. 1040, 2022.

[74] Z. Cui, T. Gao, K. Talamadupula, and Q. Ji, "Knowledge-augmented deep learning and its applications: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[75] T. R. Besold, S. Bader, H. Bowman, P. Domingos, P. Hitzler, K.-U. Kühnberger, L. C. Lamb, P. M. V. Lima, L. de Penning, G. Pinkas, *et al.*, "Neural-symbolic learning and reasoning: A survey and interpretation 1," in *Neuro-symbolic artificial intelligence: The state of the art*, pp. 1–51, IOS Press, 2021.

[76] B. P. Bhuyan, A. Ramdane-Cherif, R. Tomar, and T. Singh, "Neuro-symbolic artificial intelligence: a survey," *Neural Computing and Applications*, vol. 36, no. 21, pp. 12809–12844, 2024.

[77] W. Wang, Y. Yang, and F. Wu, "Towards data-and knowledge-driven ai: A survey on neuro-symbolic computing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 47, no. 2, pp. 878–899, 2025.

[78] X. Xie, J. Niu, X. Liu, Z. Chen, S. Tang, and S. Yu, "A survey on incorporating domain knowledge into deep learning for medical image analysis," *Medical Image Analysis*, vol. 69, p. 101985, 2021.

[79] L. Cai, J. Li, H. Lv, W. Liu, H. Niu, and Z. Wang, "Integrating domain knowledge for biomedical text analysis into deep learning: A survey," *Journal of Biomedical Informatics*, vol. 143, p. 104418, 2023.

[80] C. Meng, S. Griesemer, D. Cao, S. Seo, and Y. Liu, "When physics meets machine learning: A survey of physics-informed machine learning," *Machine Learning for Computational Science and Engineering*, vol. 1, no. 1, p. 20, 2025.

[81] L. A. Goodman, "Snowball sampling," *The Annals of Mathematical Statistics*, pp. 148–170, 1961.

[82] M. J. Page, J. E. McKenzie, P. M. Bossuyt, I. Boutron, T. C. Hoffmann, C. D. Mulrow, L. Shamseer, J. M. Tetzlaff, E. A. Akl, S. E. Brennan, *et al.*, "The prisma 2020 statement: an updated guideline for reporting systematic reviews," *bmj*, vol. 372, 2021.

[83] T. Ten Berge and R. Van Hezewijk, "Procedural and declarative knowledge: An evolutionary perspective," *Theory & Psychology*, vol. 9, no. 5, pp. 605–624, 1999.

[84] Y. Bi, "Dual coding of knowledge in the human brain," *Trends in Cognitive Sciences*, vol. 25, no. 10, pp. 883–895, 2021.

[85] E. Tulving *et al.*, "Episodic and semantic memory," *Organization of Memory*, vol. 1, no. 381-403, p. 1, 1972.

[86] L. R. Squire, "Memory systems of the brain: a brief history and current perspective," *Neurobiology of learning and memory*, vol. 82, no. 3, pp. 171–177, 2004.

[87] T. L. Griffiths, C. Kemp, and J. B. Tenenbaum, "Bayesian models of cognition," *The Cambridge Handbook of Computational Psychology*, pp. 59–100, 2001.

[88] Y. Zhang, Z. Li, M. Zhou, S. Wu, and J. Wu, "The scene language: Representing scenes with programs, words, and embeddings," in *Conference on Computer Vision and Pattern Recognition*, 2025.

[89] D. Shu, H. Zhao, X. Liu, D. Demeter, M. Du, and Y. Zhang, "Lawllm: Law large language model for the us legal system," in *Proceedings of the ACM International Conference on Information and Knowledge Management*, 2024.

[90] X. Gu, M. Chen, Y. Lin, Y. Hu, H. Zhang, C. Wan, Z. Wei, Y. Xu, and J. Wang, "On the effectiveness of large language models in domain-specific code generation," *ACM Transactions on Software Engineering and Methodology*, vol. 34, no. 3, pp. 1–22, 2025.

[91] R. K. Jones, D. Charatan, P. Guerrero, N. J. Mitra, and D. Ritchie, "Shapemod: Macro operation discovery for 3d shape programs," *ACM Transactions on Graphics*, vol. 40, no. 4, pp. 1–16, 2021.

[92] R. K. Jones, P. Guerrero, N. J. Mitra, and D. Ritchie, "Shapecoder: Discovering abstractions for visual programs from unstructured primitives," *ACM Transactions on Graphics*, vol. 42, no. 4, pp. 1–17, 2023.

[93] R. K. Jones, T. Barton, X. Xu, K. Wang, E. Jiang, P. Guerrero, N. J. Mitra, and D. Ritchie, "Shapeassembly: Learning to generate programs for 3d shape structure synthesis," *ACM Transactions on Graphics*, vol. 39, no. 6, pp. 1–20, 2020.

[94] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, I. Simon, C. Hawthorne, N. Shazeer, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, "Music transformer: Generating music with long-term structure," in *International Conference on Learning Representations*, 2019.

[95] F. Adams, A. McDannald, I. Takeuchi, and A. G. Kusne, "Human-in-the-loop for bayesian autonomous materials phase mapping," *Matter*, vol. 7, no. 2, pp. 697–709, 2024.

[96] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, *et al.*, "Highly accurate protein structure prediction with alphafold," *Nature*, vol. 596, no. 7873, pp. 583–589, 2021.

[97] Z. Mandi, Y. Weng, D. Bauer, and S. Song, "Real2code: Reconstruct articulated objects via code generation," in *International Conference on Learning Representations*, 2025.

[98] R. Calinescu, C. Paterson, and K. Johnson, "Efficient parametric model checking using domain knowledge," *IEEE Transactions on Software Engineering*, vol. 47, no. 6, pp. 1114–1133, 2019.

[99] M. Coblenz, G. Kambhatla, P. Koronkevich, J. L. Wise, C. Barnaby, J. Sunshine, J. Aldrich, and B. A. Myers, "Pliers: a process that integrates user-centered methods into programming language design," *ACM Transactions on Computer-Human Interaction (TOCHI)*, vol. 28, no. 4, pp. 1–53, 2021.

[100] M. Edmonds, F. Gao, H. Liu, X. Xie, S. Qi, B. Rothrock, Y. Zhu, Y. N. Wu, H. Lu, and S.-C. Zhu, "A tale of two explanations: Enhancing human trust by explaining robot behavior," *Science Robotics*, vol. 4, no. 37, p. eaay4663, 2019.

[101] W. Zhong, Z. Yang, and C. Y.-C. Chen, "Retrosynthesis prediction using an end-to-end graph generative architecture for molecular graph editing," *Nature Communications*, vol. 14, no. 1, p. 3009, 2023.

[102] X. Xie, X. Tang, S. Gu, and L. Cui, "An intelligent guided troubleshooting method for aircraft based on hybirdrag," *Scientific Reports*, vol. 15, no. 1, pp. 1–16, 2025.

[103] Y. Ge, W. Hua, K. Mei, J. Tan, S. Xu, Z. Li, Y. Zhang, *et al.*, "Openagi: When llm meets domain experts," in *Advances in Neural Information Processing Systems*, 2023.

[104] Z. Huang, W. Zhao, J. Gao, Z. Hu, X. Luo, Y. Cao, Y. Chen, Y. Sun, and W. Wang, "Physics-informed regularization for domain-agnostic dynamical system modeling," in *Advances in Neural Information Processing Systems*, 2024.

[105] R. Yen, J. S. Zhu, S. Suh, H. Xia, and J. Zhao, "Coladder: Manipulating code generation via multi-level blocks," in *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*, 2024.

[106] K. Ellis, A. Albright, A. Solar-Lezama, J. B. Tenenbaum, and T. J. O'Donnell, "Synthesizing theories of human language with bayesian program induction," *Nature Communications*, vol. 13, no. 1, p. 5024, 2022.

[107] Ö. Babur, A. Luna, A. Korkut, F. Durupinar, M. C. Siper, U. Dogrusoz, A. S. V. Jacome, R. Peckner, K. E. Christianson, J. D. Jaffe, *et al.*, "Causal interactions from proteomic profiles: Molecular data meet pathway knowledge," *Patterns*, vol. 2, no. 6, 2021.

[108] S. Hou, Z. Shen, A. Zhao, J. Liang, Z. Gui, X. Guan, R. Li, and H. Wu, "Geocode-gpt: A large language model for geospatial code generation," *International Journal of Applied Earth Observation and Geoinformation*, p. 104456, 2025.

[109] C. Jiang, S. Qi, Y. Zhu, S. Huang, J. Lin, L.-F. Yu, D. Terzopoulos, and S.-C. Zhu, "Configurable 3d scene synthesis and 2d image rendering with per-pixel ground truth using stochastic grammars," *International Journal of Computer Vision*, vol. 126, no. 9, pp. 920–941, 2018.

[110] M. Yang, P. Zhu, J. Cheema, R. Bloomer, P. Mikulski, Q. Liu, Y. Zhang, C. Dean, and Y. Ding, "In vivo single-molecule analysis reveals coolair rna structural diversity," *Nature*, vol. 609, no. 7926, pp. 394–399, 2022.

[111] H. Zhang, L. Zhang, A. Lin, C. Xu, Z. Li, K. Liu, B. Liu, X. Ma, F. Zhao, H. Jiang, *et al.*, "Algorithm for optimized mrna design improves stability and immunogenicity," *Nature*, vol. 621, no. 7978, pp. 396–403, 2023.

[112] Z. Lai, K. Wei, Y. Fu, P. Härtel, and F. Heide, "∇-prox: Differentiable proximal algorithm modeling for large-scale optimization," *ACM Transactions on Graphics*, vol. 42, no. 4, pp. 1–19, 2023.

[113] L. Chen, Y. Zhang, Y. Song, Z. Shen, and L. Liu, "Lfme: A simple framework for learning from multiple experts in domain generalization," in *Advances in Neural Information Processing Systems*, 2024.

[114] K. A. Bozhüyük, L. Präve, C. Kegler, L. Schenk, S. Kaiser, C. Schelhas, Y.-N. Shi, W. Kuttenlochner, M. Schreiber, J. Kandler, *et al.*, "Evolution-inspired engineering of nonribosomal peptide synthetases," *Science*, vol. 383, no. 6689, p. eadg4320, 2024.

[115] E. Qu and A. Krishnapriyan, "The importance of being scalable: Improving the speed and accuracy of neural network interatomic potentials across chemical domains," in *Advances in Neural Information Processing Systems*, 2024.

[116] M. Abdallah, V. Iovene, G. Zanitti, and D. Wassermann, "Meta-analysis of the functional neuroimaging literature with probabilistic logic programming," *Scientific Reports*, vol. 12, no. 1, p. 19431, 2022.

[117] Y.-L. Li, X. Liu, X. Wu, Y. Li, Z. Qiu, L. Xu, Y. Xu, H.-S. Fang, and C. Lu, "Hake: A knowledge engine foundation for human activity understanding," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 7, pp. 8494–8506, 2022.

[118] S. Tan, M. Ge, D. Guo, H. Liu, and F. Sun, "Knowledge-based embodied question answering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 10, pp. 11948–11960, 2023.

[119] C. Wang, J. Thompson, and B. Lee, "Data formulator: Ai-powered concept-driven visualization authoring," *IEEE Transactions on Visualization and Computer Graphics*, vol. 30, no. 1, pp. 1128–1138, 2023.

[120] Y. Zhang, Y. Cao, X. Xu, and W. Shen, "Logicode: an llm-driven framework for logical anomaly detection," *IEEE Transactions on Automation Science and Engineering*, 2024.

[121] H. W. Wang, M. Gordon, L. Battle, and J. Heer, "Dracogpt: Extracting visualization design preferences from large language models," *IEEE Transactions on Visualization and Computer Graphics*, 2024.

[122] S. Zhang, Y. Wang, X. Liu, and Z. Ji, "Operation analysis and fault path recognition of complex industrial systems based on boolean networks," *IEEE Transactions on Automation Science and Engineering*, 2025.

[123] W. Liu, J. Li, Y. Tang, Y. Zhao, C. Liu, M. Song, Z. Ju, S. V. Kumar, Y. Lu, R. Akbani, *et al.*, "Drbioright 2.0: an llm-powered bioinformatics chatbot for large-scale cancer functional proteomics analysis," *Nature Communications*, vol. 16, no. 1, p. 2256, 2025.

[124] J. Yao, X. Wang, J. Lian, X. Yi, and X. Xie, "Neural recommendation reasoning with logic rules," *ACM Transactions on Information Systems*, 2025.

[125] M. Hu, M. Zhang, F. Mallet, X. Fu, and M. Chen, "Accelerating reinforcement learning-based ccsl specification synthesis using curiosity-driven exploration," *IEEE Transactions on Computers*, vol. 72, no. 5, pp. 1431–1446, 2022.

[126] R. Rauschen, M. Guy, J. E. Hein, and L. Cronin, "Universal chemical programming language for robotic synthesis repeatability," *Nature Synthesis*, vol. 3, no. 4, pp. 488–496, 2024.

[127] A. Zirak and H. Hemmati, "Improving automated program repair with domain adaptation," *ACM Transactions on Software Engineering and Methodology*, vol. 33, no. 3, pp. 1–43, 2024.

[128] J. Dagdelen, A. Dunn, S. Lee, N. Walker, A. S. Rosen, G. Ceder, K. A. Persson, and A. Jain, "Structured information extraction from scientific text with large language models," *Nature Communications*, vol. 15, no. 1, p. 1418, 2024.

[129] Q. Yang, H. Zuo, R. Su, H. Su, T. Zeng, H. Zhou, R. Wang, J. Chen, Y. Lin, Z. Chen, *et al.*, "Dual retrieving and ranking medical large language model with retrieval augmented generation," *Scientific Reports*, vol. 15, no. 1, pp. 1–10, 2025.

[130] A. Zhao, J. Xu, M. Konaković-Luković, J. Hughes, A. Spielberg, D. Rus, and W. Matusik, "Robogrammar: graph grammar for terrain-optimized robot design," *ACM Transactions on Graphics*, vol. 39, no. 6, pp. 1–16, 2020.

[131] P. Merrell, "Example-based procedural modeling using graph grammars," *ACM Transactions on Graphics*, vol. 42, no. 4, pp. 1–16, 2023.

[132] B. Wang, Z. Wang, X. Wang, Y. Cao, R. A Saurous, and Y. Kim, "Grammar prompting for domain-specific language generation with large language models," in *Advances in Neural Information Processing Systems*, 2023.

[133] L. Ai, S.-S. Liang, W.-Z. Dai, L. Hallett, S. H. Muggleton, and G. S. Baldwin, "Human comprehensible active learning of genome-scale metabolic networks," in *AAAI Spring Symposium Series*, 2023.

[134] X. Yao, H. Li, T. H. Chan, W. Xiao, M. Yuan, Y. Huang, L. Chen, and B. Yu, "Hdldebugger: Streamlining hdl debugging with large language models," *ACM Transactions on Design Automation of Electronic Systems*, 2024.

[135] K. Alrashedy, P. Tambwekar, Z. H. Zaidi, M. Langwasser, W. Xu, and M. Gombolay, "Generating cad code with vision-language models for 3d designs," in *International Conference on Learning Representations*, 2025.

[136] K. Ye, W. Ni, M. Krieger, D. Ma'ayan, J. Wise, J. Aldrich, J. Sunshine, and K. Crane, "Penrose: from mathematical notation to beautiful diagrams," *ACM Transactions on Graphics*, vol. 39, no. 4, pp. 144–1, 2020.

[137] G. Li and X. Yuan, "Gotreescape: Navigate and explore the tree visualization design space," *IEEE Transactions on Visualization and Computer Graphics*, vol. 29, no. 12, pp. 5451–5467, 2022.

[138] R. Yuan, H. Lin, Y. Wang, Z. Tian, S. Wu, T. Shen, G. Zhang, Y. Wu, C. Liu, Z. Zhou, *et al.*, "Chatmusician: Understanding and generating music intrinsically with llm," in *Findings of the Association for Computational Linguistics*, 2024.

[139] A. Mavrogiannis, C. Mavrogiannis, and Y. Aloimonos, "Cook2ltl: Translating cooking recipes to ltl formulae using large language models," in *IEEE International Conference on Robotics and Automation*, 2024.

[140] M. Liu, Y.-D. Tsai, W. Zhou, and H. Ren, "Craftrtl: High-quality synthetic data generation for verilog code models with correct-by-construction non-textual representations and targeted code repair," in *International Conference on Learning Representations*, 2025.

[141] T. Du, J. P. Inala, Y. Pu, A. Spielberg, A. Schulz, D. Rus, A. Solar-Lezama, and W. Matusik, "Inversecsg: Automatic conversion of 3d models to csg trees," *ACM Transactions on Graphics*, vol. 37, no. 6, pp. 1–16, 2018.

[142] L. Fish, M. Khoroshkin, A. Navickas, K. Garcia, B. Culbertson, B. Hänisch, S. Zhang, H. C. Nguyen, L. M. Soto, M. Dermit, *et al.*, "A prometastatic splicing program regulated by snrpa1 interactions with structured rna elements," *Science*, vol. 372, no. 6543, p. eabc7531, 2021.

[143] X. Zhang, Y. Chen, M. Tang, Z. Lei, and J. Wang, "Grammar-induced wavelet network for human parsing," *IEEE Transactions on Image Processing*, vol. 31, pp. 4502–4514, 2022.

[144] Y. Zhang, X. Li, H. Xie, W. Zhuang, S. Guo, and Z. Li, "Multi-label action anticipation for real-world videos with scene understanding," *IEEE Transactions on Image Processing*, 2024.

[145] M. Lázaro-Gredilla, D. Lin, J. S. Guntupalli, and D. George, "Beyond imitation: Zero-shot task transfer on robots by learning concepts as cognitive programs," *Science Robotics*, vol. 4, no. 26, p. eaav3150, 2019.

[146] Y.-X. Huang, W.-Z. Dai, J. Yang, L.-W. Cai, S. Cheng, R. Huang, Y.-F. Li, and Z.-H. Zhou, "Semi-supervised abductive learning and its application to theft judicial sentencing," in *IEEE International Conference on Data Mining (ICDM)*, 2020.

[147] N. H. Park, M. Manica, J. Born, J. L. Hedrick, T. Erdmann, D. Y. Zubarev, N. Adell-Mill, and P. L. Arrechea, "Artificial intelligence driven design of catalysts and materials for ring opening polymerization using a domain-specific language," *Nature Communications*, vol. 14, no. 1, p. 3686, 2023.

[148] E.-H. Gao, Y.-X. Huang, W.-C. Hu, X.-H. Zhu, and W.-Z. Dai, "Knowledge-enhanced historical document segmentation and recognition," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.

[149] M. X. Liu, F. Liu, A. J. Fiannaca, T. Koo, L. Dixon, M. Terry, and C. J. Cai, ""we need structured output": Towards user-centered constraints on large language model output," in *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, 2024.

[150] S. H. M. Mehr, M. Craven, A. I. Leonov, G. Keenan, and L. Cronin, "A universal system for digitization and automatic execution of the chemical synthesis literature," *Science*, vol. 370, no. 6512, pp. 101–108, 2020.

[151] S. Rohrbach, M. Šiaučiulis, G. Chisholm, P.-A. Pirvan, M. Saleeb, S. H. M. Mehr, E. Trushina, A. I. Leonov, G. Keenan, A. Khan, *et al.*, "Digitization and validation of a chemical synthesis literature database in the chempu," *Science*, vol. 377, no. 6602, pp. 172–180, 2022.

[152] B. Romera-Paredes, M. Barekatain, A. Novikov, M. Balog, M. P. Kumar, E. Dupont, F. J. Ruiz, J. S. Ellenberg, P. Wang, O. Fawzi, *et al.*, "Mathematical discoveries from program search with large language models," *Nature*, vol. 625, no. 7995, pp. 468–475, 2024.

[153] X. Tang, B. Qian, R. Gao, J. Chen, X. Chen, and M. B. Gerstein, "Biocoder: a benchmark for bioinformatics code generation with large language models," *Bioinformatics*, vol. 40, no. Supplement_1, pp. i266–i276, 2024.

[154] Y. Li, H. Pei, L. Huang, B. Yin, and K.-Y. Cai, "Automatic repair of quantum programs via unitary operation," *ACM Transactions on Software Engineering and Methodology*, vol. 33, no. 6, pp. 1–43, 2024.

[155] A. Shaikhha, M. Elseidy, S. Mihaila, D. Espino, and C. Koch, "Synthesis of incremental linear algebra programs," *ACM Transactions on Database Systems (TODS)*, vol. 45, no. 3, pp. 1–44, 2020.

[156] Q. Chen, Y. Chen, R. Zou, W. Shuai, Y. Guo, J. Wang, and N. Cao, "Chart2vec: A universal embedding of context-aware visualizations," *IEEE Transactions on Visualization and Computer Graphics*, 2024.

[157] O. O'Donoghue, A. Shtedritski, J. Ginger, R. Abboud, A. Ghareeb, and S. Rodriques, "Bioplanner: Automatic evaluation of llms on protocol planning in biology," in *Annual Conference on Empirical Methods in Natural Language Processing*, 2023.

[158] D. Carmona and H. Yu, "Bicap: A novel bi-modal dataset of daily living dual-arm manipulation actions," *The International Journal of Robotics Research*, vol. 44, no. 6, pp. 891–907, 2025.

[159] A. Raistrick, L. Lipson, Z. Ma, L. Mei, M. Wang, Y. Zuo, K. Kayan, H. Wen, B. Han, Y. Wang, *et al.*, "Infinite photorealistic worlds using procedural generation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12630–12641, 2023.

[160] A. Raistrick, L. Mei, K. Kayan, D. Yan, Y. Zuo, B. Han, H. Wen, M. Parakh, S. Alexandropoulos, L. Lipson, *et al.*, "Infinigen indoors: Photorealistic indoor scenes using procedural generation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[161] C. Li, H. Pan, A. Bousseau, and N. J. Mitra, "Free2cad: Parsing free-hand drawings into cad commands," *ACM Transactions on Graphics*, vol. 41, no. 4, pp. 1–16, 2022.

[162] J. Q. Wu, N. Horeweg, M. de Bruyn, R. A. Nout, I. M. Jürgenliemk-Schulz, L. C. Lutgens, J. J. Jobsen, E. M. Van der Steen-Banasik, H. W. Nijman, V. T. Smit, *et al.*, "Automated causal inference in application to randomized controlled clinical trials," *Nature Machine Intelligence*, vol. 4, no. 5, pp. 436–444, 2022.

[163] S. Yang, C. Dong, Y. Xiao, Y. Cheng, Z. Shi, Z. Li, and L. Sun, "Asteria-pro: Enhancing deep learning-based binary code similarity detection by incorporating domain knowledge," *ACM Transactions on Software Engineering and Methodology*, vol. 33, no. 1, pp. 1–40, 2023.

[164] N. Kikkawa and H. Ohno, "Materials discovery using max k-armed bandit," *Journal of Machine Learning Research*, vol. 25, no. 100, pp. 1–40, 2024.

[165] G. Davidson, G. Todd, J. Togelius, T. M. Gureckis, and B. M. Lake, "Goals as reward-producing programs," *Nature Machine Intelligence*, vol. 7, no. 2, pp. 205–220, 2025.

[166] L. Wu, K. Wang, K. Nie, S. Guo, C. Gao, Z. Wang, and S. Li, "Tfgin: Tight-fitting graph inference network for table-based fact verification," *ACM Transactions on Information Systems*, 2025.

[167] S. Qi, B. Jia, S. Huang, P. Wei, and S.-C. Zhu, "A generalized earley parser for human activity parsing and prediction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 8, pp. 2538–2554, 2020.

[168] J. Belouadi, S. Ponzetto, and S. Eger, "Detikzify: Synthesizing graphics programs for scientific figures and sketches with tikz," in *Advances in Neural Information Processing Systems*, 2024.

[169] C. K. Williams, "Structured generative models for scene understanding," *International Journal of Computer Vision*, vol. 133, no. 5, pp. 2845–2867, 2025.

[170] M. H. Segler, M. Preuss, and M. P. Waller, "Planning chemical syntheses with deep neural networks and symbolic ai," *Nature*, vol. 555, no. 7698, pp. 604–610, 2018.

[171] R. Lioutikov, G. Maeda, F. Veiga, K. Kersting, and J. Peters, "Learning attribute grammars for movement primitive sequencing," *The International Journal of Robotics Research*, vol. 39, no. 1, pp. 21–38, 2020.

[172] Y. Pane, M. H. Arbo, E. Aertbeliën, and W. Decré, "A system architecture for cad-based robotic assembly with sensor-based skills," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 3, pp. 1237–1249, 2020.

[173] G. Coruhlu, E. Erdem, and V. Patoglu, "Explainable robotic plan execution monitoring under partial observability," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2495–2515, 2021.

[174] Y. Yamagata, T. Fukuyama, S. Onami, and H. Masuya, "Prototyping an ontological framework for cellular senescence mechanisms: A homeostasis imbalance perspective," *Scientific Data*, vol. 11, no. 1, p. 485, 2024.

[175] Z. Wang, S. Cai, A. Liu, Y. Jin, J. Hou, B. Zhang, H. Lin, Z. He, Z. Zheng, Y. Yang, *et al.*, "Jarvis-1: Open-world multi-task agents with memory-augmented multimodal language models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[176] J. Qi, Z. Luan, S. Huang, C. Fung, and H. Yang, "Logsay: An efficient comprehension system for log numerical reasoning," *IEEE Transactions on Computers*, 2024.

[177] J. G. Greener, L. Moffat, and D. T. Jones, "Design of metalloproteins and novel protein folds using variational autoencoders," *Scientific Reports*, vol. 8, no. 1, p. 16189, 2018.

[178] W. Sun, Y. Zheng, K. Yang, Q. Zhang, A. A. Shah, Z. Wu, Y. Sun, L. Feng, D. Chen, Z. Xiao, *et al.*, "Machine learning-assisted molecular design and efficiency prediction for high-performance organic photovoltaic materials," *Science Advances*, vol. 5, no. 11, p. eaay4275, 2019.

[179] A. A. Soares, P. V. Martins, and A. R. da Silva, "Legallanguage: A domain-specific language for legal contexts," in *Enterprise Engineering Working Conference*, 2019.

[180] K. D. Willis, Y. Pu, J. Luo, H. Chu, T. Du, J. G. Lambourne, A. Solar-Lezama, and W. Matusik, "Fusion 360 gallery: A dataset and environment for programmatic cad construction from human design sequences," *ACM Transactions on Graphics*, vol. 40, no. 4, pp. 1–24, 2021.

[181] D. A. Boiko, R. MacKnight, B. Kline, and G. Gomes, "Autonomous chemical research with large language models," *Nature*, vol. 624, no. 7992, pp. 570–578, 2023.

[182] D. Slack, S. Krishna, H. Lakkaraju, and S. Singh, "Explaining machine learning models with interactive natural language conversations using talktomodel," *Nature Machine Intelligence*, vol. 5, no. 8, pp. 873–883, 2023.

[183] H. Zhu, S. Wilson, and E. Feron, "The design, education and evolution of a robotic baby," *IEEE Transactions on Robotics*, vol. 39, no. 3, pp. 2488–2507, 2023.

[184] R. Roy and H. M. Al-Hashimi, "Alphafold3 takes a step toward decoding molecular behavior and biological computation," *Nature Structural & Molecular Biology*, vol. 31, no. 7, pp. 997–1000, 2024.

[185] P. Colombo, T. Pires, M. Boudiaf, R. Melo, D. Culver, E. Malaboeuf, G. Hautreux, J. Charpentier, and M. Desa, "Saullm-54b & saullm-141b: Scaling up domain adaptation for the legal domain," in *Advances in Neural Information Processing Systems*, 2024.

[186] Y. Zheng, H. Y. Koh, J. Ju, A. T. Nguyen, L. T. May, G. I. Webb, and S. Pan, "Large language models for scientific discovery in molecular property prediction," *Nature Machine Intelligence*, pp. 1–11, 2025.

[187] B. Meyers, H. Vangheluwe, J. Denil, and R. Salay, "A framework for temporal verification support in domain-specific modelling," *IEEE Transactions on Software Engineering*, vol. 46, no. 4, pp. 362–404, 2018.

[188] A. Carcangiu, L. D. Spano, G. Fumera, and F. Roli, "Deictic: A compositional and declarative gesture description based on hidden markov models," *International Journal of Human-Computer Studies*, vol. 122, pp. 113–132, 2019.

[189] E. Berger, D. Yorukoglu, L. Zhang, S. K. Nyquist, A. K. Shalek, M. Kellis, I. Numanagić, and B. Berger, "Improved haplotype inference by exploiting long-range linking and allelic imbalance in rna-seq datasets," *Nature Communications*, vol. 11, no. 1, p. 4662, 2020.

[190] N. Ritschel, V. Kovalenko, R. Holmes, R. Garcia, and D. C. Shepherd, "Comparing block-based programming models for two-armed robots," *IEEE Transactions on Software Engineering*, vol. 48, no. 5, pp. 1630–1643, 2020.

[191] S. Lyi, Q. Wang, F. Lekschas, and N. Gehlenborg, "Gosling: A grammar-based toolkit for scalable and interactive genomics data visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 1, pp. 140–150, 2021.

[192] J. Mao, T. Lozano-Pérez, J. Tenenbaum, and L. Kaelbling, "Pdsketch: Integrated domain programming, learning, and planning," in *Advances in Neural Information Processing Systems*, 2022.

[193] N. Catano, "Program synthesis for cyber-resilience," *IEEE Transactions on Software Engineering*, vol. 49, no. 3, pp. 962–972, 2022.

[194] W. Wan, F. Zhang, C. Zhang, M. Zhang, J. Zhai, Y. Chai, H. Zhang, W. Lu, Y. Chen, H. Li, *et al.*, "Compressed data direct computing for databases," *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, no. 5, pp. 1902–1918, 2023.

[195] T. Chen, Q. Wang, Z. Dong, L. Shen, and X. Peng, "Enhancing robot program synthesis through environmental context," in *Advances in Neural Information Processing Systems*, 2023.

[196] S. Wang, H. Yan, K. E. Isaacs, and Y. Sun, "Visual exploratory analysis for designing large-scale network-on-chip architectures: A domain expert-led design study," *IEEE Transactions on Visualization and Computer Graphics*, vol. 30, no. 4, pp. 1970–1983, 2023.

[197] Z.-S. Chen, M.-D. Zhou, K.-S. Chin, A. Darko, X.-J. Wang, and W. Pedrycz, "Optimized decision support for bim maturity assessment," *Automation in Construction*, vol. 149, p. 104808, 2023.

[198] M. Korosteleva and O. Sorkine-Hornung, "Garmentcode: Programming parametric sewing patterns," *ACM Transactions on Graphics*, vol. 42, no. 6, pp. 1–15, 2023.

[199] R. K. Jones, R. Zhang, A. Ganeshan, and D. Ritchie, "Learning to edit visual programs with self-supervision," in *Advances in Neural Information Processing Systems*, 2024.

[200] Z. Guo, A. Kale, M. Kay, and J. Hullman, "Vmc: A grammar for visualizing statistical model checks," *IEEE Transactions on Visualization and Computer Graphics*, 2024.

[201] Y. Yan, N. Cooper, K. Moran, G. Bavota, D. Poshyvanyk, and S. Rich, "Enhancing code understanding for impact analysis by combining transformers and program dependence graphs," *Proceedings of the ACM on Software Engineering*, vol. 1, no. FSE, pp. 972–995, 2024.

[202] B. J. Oakes, M. Famelis, and H. Sahraoui, "Building domain-specific machine learning workflows: A conceptual framework for the state of the practice," *ACM Transactions on Software Engineering and Methodology*, vol. 33, no. 4, pp. 1–50, 2024.

[203] Y. Gui, Y. Wan, Z. Li, Z. Zhang, D. Chen, H. Zhang, Y. Su, B. Chen, X. Zhou, W. Jiang, *et al.*, "Uicopilot: Automating ui synthesis via hierarchical code generation from webpage designs," in *Proceedings of the ACM on Web Conference*, 2025.

[204] R. Mon-Williams, G. Li, R. Long, W. Du, and C. G. Lucas, "Embodied large language models enable robots to complete complex tasks in unpredictable environments," *Nature Machine Intelligence*, pp. 1–10, 2025.

[205] D. Marr, *Vision: A computational investigation into the human representation and processing of visual information*. W. H. Freeman, San Francisco, CA, 1982.

[206] B. S. Blanchard, W. J. Fabrycky, and W. J. Fabrycky, *Systems engineering and analysis*. Prentice hall Englewood Cliffs, NJ, 1990.

[207] U. Frank, "Domain-specific modeling languages: requirements analysis and design guidelines," *Domain engineering: Product lines, languages, and conceptual models*, pp. 133–157, 2013.

[208] A. M. McNutt, "No grammar to rule them all: A survey of json-style dsls for visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 29, no. 1, pp. 160–170, 2022.

[209] Y. Wan, Z. Bi, Y. He, J. Zhang, H. Zhang, Y. Sui, G. Xu, H. Jin, and P. Yu, "Deep learning for code intelligence: Survey, benchmark and toolkit," *ACM Computing Surveys*, vol. 56, no. 12, pp. 1–41, 2024.

[210] M. R. Munafò, B. A. Nosek, D. V. Bishop, K. S. Button, C. D. Chambers, N. Percie du Sert, U. Simonsohn, E.-J. Wagenmakers, J. J. Ware, and J. Ioannidis, "A manifesto for reproducible science," *Nature Human Behaviour*, vol. 1, no. 1, pp. 1–9, 2017.

[211] Y.-Z. Shi, H. Hou, Z. Bi, F. Meng, X. Wei, L. Ruan, and Q. Wang, "AutoDSL: Automated domain-specific language design for structural representation of procedures with constraints," in *Annual Meeting of the Association for Computational Linguistics*, 2024.

[212] Y.-Z. Shi, M. Liu, H. Ma, Q. Xu, H. Qu, K. He, L. Ruan, and Q. Wang, "Targeted control of fast prototyping through domain-specific interface," in *International Conference on Machine Learning*, 2025.

[213] Y.-Z. Shi, M. Liu, F. Meng, Q. Xu, Z. Bi, K. He, L. Ruan, and Q. Wang, "Hierarchically Encapsulated Representation for Protocol Design in Self-Driving Labs," in *International Conference on Learning Representations*, 2025.

[214] Y.-Z. Shi, Q. Xu, Y. Li, M. Liu, H. Qu, L. Ruan, and Q. Wang, "Automated constraint specification for job scheduling by regulating generative model with domain-specific representation," *IEEE Transactions on Automation Science and Engineering*, 2025.

[215] Y.-Z. Shi, H. Li, L. Ruan, and H. Qu, "Constraint representation towards precise data-driven storytelling," in *IEEE Visualization and Visual Analytics Gen4DS Workshop*, 2024.

[216] Y.-Z. Shi, F. Meng, H. Hou, Z. Bi, Q. Xu, L. Ruan, and Q. Wang, "Expert-level protocol translation for self-driving labs," in *Advances in Neural Information Processing Systems*, 2024.

[217] Y.-Z. Shi, Q. Xu, F. Meng, L. Ruan, and Q. Wang, "Abstract Hardware Grounding towards the Automated Design of Automation Systems," in *International Conference on Intelligent Robotics and Applications*, 2024.

[218] Y.-Z. Shi, S. Li, X. Niu, Q. Xu, J. Liu, Y. Xu, S. Gu, B. He, X. Li, X. Zhao, *et al.*, "PersLEARN: Research Training through the Lens of Perspective Cultivation," in *Annual Meeting of the Association for Computational Linguistics*, 2023.

[219] Y.-Z. Shi, M. Xu, J. E. Hopcroft, K. He, J. B. Tenenbaum, S.-C. Zhu, Y. N. Wu, W. Han, and Y. Zhu, "On the complexity of Bayesian generalization," in *International Conference on Machine Learning*, 2023.

[220] D. Bjørner, "Domain engineering," in *Formal Methods: State of the Art and New Directions*, pp. 1–41, Springer, 2009.

[221] G. Guizzardi, "Ontology-based evaluation and design of visual conceptual modeling languages," *Domain Engineering: Product Lines, Languages, and Conceptual Models*, pp. 317–347, 2013.

[222] G. Karsai, H. Krahn, C. Pinkernell, B. Rumpe, M. Schindler, and S. Völkel, "Design guidelines for domain specific languages," in *OOPSLA Workshop on Domain-Specific Modeling*, 2009.
[223] J. B. Bernerth and H. Aguinis, "A critical review and best-practice recommendations for control variable usage," *Personnel Psychology*, vol. 69, no. 1, pp. 229–283, 2016.
[224] F. Bento, M. Tagliabue, and F. Lorenzo, "Organizational silos: A scoping review informed by a behavioral perspective on systems and networks," *Societies*, vol. 10, no. 3, p. 56, 2020.
[225] G. Tett, *The silo effect: The peril of expertise and the promise of breaking down barriers*. Simon and Schuster, 2016.
[226] Y. Rizk, M. Awad, and E. W. Tunstel, "Cooperative heterogeneous multi-robot systems: A survey," *ACM Computing Surveys*, vol. 52, no. 2, pp. 1–31, 2019.
[227] A. Schieppati, J.-I. Henter, E. Daina, and A. Aperia, "Why rare diseases are an important medical and social issue," *The Lancet*, vol. 371, no. 9629, pp. 2039–2041, 2008.
[228] M. Felleisen, "On the expressive power of programming languages," *Science of Computer Programming*, vol. 17, no. 1-3, pp. 35–75, 1991.
[229] S. Reed, K. Zolna, E. Parisotto, S. G. Colmenarejo, A. Novikov, G. Barth-maron, M. Giménez, Y. Sulsky, J. Kay, J. T. Springenberg, *et al.*, "A generalist agent," *Transactions on Machine Learning Research*, 2022.
[230] T. Fukuda, P. Dario, and G.-Z. Yang, "Humanoid robotics—history, current state of the art, and challenges," *Science Robotics*, vol. 2, no. 13, p. eaar4043, 2017.
[231] T. Tu, S. Azizi, D. Driess, M. Schaekermann, M. Amin, P.-C. Chang, A. Carroll, C. Lau, R. Tanno, I. Ktena, *et al.*, "Towards generalist biomedical ai," *NEJM AI*, vol. 1, no. 3, p. AIoa2300138, 2024.
[232] A. Gupta, S. Savarese, S. Ganguli, and L. Fei-Fei, "Embodied intelligence via learning and evolution," *Nature Communications*, vol. 12, no. 1, p. 5721, 2021.
[233] O. Shoval, H. Sheftel, G. Shinar, Y. Hart, O. Ramote, A. Mayo, E. Dekel, K. Kavanagh, and U. Alon, "Evolutionary trade-offs, pareto optimality, and the geometry of phenotype space," *Science*, vol. 336, no. 6085, pp. 1157–1160, 2012.



Yu-Zhe Shi is presently a Ph.D. Candidate in the Department of Computer Science and Engineering at the Hong Kong University of Science and Technology. His research primarily focuses on the automated design, deployment, and communication of domain-specific representations, aiming to facilitate interaction, collaboration, coordination, and communication among humans and machines.



Zhangqian Bi is currently a Ph.D. candidate in the School of Computer Science and Technology at Huazhong University of Science and Technology (HUST), China. He received his B.S. degree in Computer Science and Technology from HUST. His research interests include program analysis and synthesis, artificial intelligence for software engineering and interdisciplinary applications.



Lecheng Ruan received the B.S. honor degree from the School of Mechatronic Engineering, Harbin Institute of Technology in 2015, and the Ph.D. degree from the Department of Mechanical and Aerospace Engineering, University of California, Los Angeles in 2020. He is now directing the Center of Intelligence in the Laboratory of Human-Robot Systems, Peking University. His research interests include perception, control and knowledge representation of robotic systems.



Yao Wan is currently an Associate Professor in the College of Computer Science and Technology at Huazhong University of Science and Technology, Wuhan, China. He received his Ph.D. degree from Zhejiang University in 2019. His research interests lie at the intersection of software engineering and artificial intelligence, with a particular focus on programming languages, natural language processing, and large language models.



Huamin Qu is a Chair Professor in the Department of Computer Science and Engineering at Hong Kong University of Science and Technology (HKUST) and also the Dean of the Academy of Interdisciplinary Studies of HKUST. He obtained a BS in Mathematics from Xi'an Jiaotong University, China, an MS and a PhD in Computer Science from the Stony Brook University. His main research interests are in visualization and human-computer interaction, with focuses on urban informatics, E-learning, and explainable artificial intelligence.



Qining Wang (Senior Member, IEEE) received his Ph.D. degree from Peking University, Beijing, China, in 2009, where he is currently Dean and Full Professor in the School of Advanced Manufacturing and Robotics, Peking University. He has authored/co-authored over 200 scientific papers in international journals and refereed conference proceedings. His research interests include wearable robotics and human-machine interfaces.



Philip S. Yu (Life Fellow, IEEE) is a Distinguished Professor in Computer Science at the University of Illinois Chicago and also holds the Wexler Chair in Information Technology. Dr. Yu spent most of his career at IBM, where he was manager of the Software Tools and Techniques group at the Watson Research Center. His research interest is on big data, including data mining, data stream, database and privacy. He has published more than 920 papers in refereed journals and conferences. He holds or has applied for more than 250 US patents. Dr. Yu is a Fellow of the ACM and the IEEE. He is the Editor-in-Chief of ACM Transactions on Knowledge Discovery from Data. He is on the steering committee of the IEEE Conference on Data Mining and ACM Conference on Information and Knowledge Management and was a member of the IEEE Data Engineering steering committee. He was the Editor-in-Chief of IEEE Transactions on Knowledge and Data Engineering (2001-2004).