

Enduring Strengths of Knowledge-Based Methods in the Era of Machine Learning and Large Language Models: A Task-and-Architecture Perspective

MAIKEL LEON

Department of Business Technology
University of Miami
Miami, Florida,
USA

Abstract: Large language models have shifted AI toward statistical learning, but knowledge-based methods remain essential for tasks governed by combinatorial structure, declarative correctness, strong domain priors, and auditable reasoning. This paper treats the issue as one of task–architecture fit rather than paradigm competition. It surveys evidence across twelve problem classes, from satisfiability, constraint solving, planning, and formal mathematics to formal argumentation, answer set programming, knowledge graphs, and interpretable rule- and concept-based models, reporting concrete benchmark results instead of broad claims of superiority. From this evidence, the paper proposes nine task characteristics that predict when symbolic, neural, or hybrid approaches are most appropriate, connects them to the major symbolic traditions, and reviews integration patterns such as logic regularization, constraint-guided decoding, search–verification loops, graph retrieval, and probabilistic logic programming. Beyond the survey, an original formal analysis derives error propagation bounds for search–verification loops, convergence conditions for logic regularization, complexity results for constraint-guided decoding across constraint families, and a unified cost-accuracy model for hybrid architectures, and states two characteristic failure modes of hybrid systems as formal propositions. The central conclusion is that robust AI systems increasingly depend on principled hybrids that combine neural flexibility with explicit knowledge and verifiable reasoning.

Key-Words: Symbolic AI; Neuro-symbolic Integration; Large Language Models; Knowledge Graphs; Formal Reasoning; Explainable AI.

Received: March 29, 2026. Revised: June 15, 2026. Accepted: July 16, 2026. Published: September 23, 2026.

1 Introduction

Since 2020, large language models (LLMs) have moved from research curiosity to everyday infrastructure. They are not narrow classifiers but generative sequence models that, at sufficient scale and with appropriate instruction tuning, exhibit broad capabilities across natural-language understanding, code generation, retrieval, tool use, common-sense association, and approximate reasoning, [1], [2]. For tasks dominated by perception, dialogue, summarization, or fluent code drafting, LLMs and other large neural models are now state-of-the-art and deployed at scale. Any accurate contemporary account of AI must begin from that fact.

Yet careful empirical studies, particularly in planning, formal verification, and combinatorial optimization, show that purely autoregressive systems remain unreliable when problems demand correctness with respect to an explicit specification, completeness over a search space, or transfer beyond the training distribution’s support. A sequence of position papers grounded in benchmark results argues

that autoregressive LLMs cannot, by themselves, plan or self-verify reliably, and that dependable planning capability comes from coupling LLMs with external symbolic verifiers in what has been termed an “LLM-Modulo” configuration in [3], [4], [5]. Independent evaluations confirm the pattern: chain-of-thought prompting on reasoning and planning benchmarks is highly sensitive to prompt construction and tends to degrade as problems scale or drift from in-context examples, [6], [7], [8]. A separate line of work makes broader cases for symbolic structure in robust AI and articulates explicit neuro-symbolic research agendas; see, for example, [9], [10], [11] and the more recent task-directed survey in [12]. These positions are not interchangeable: some authors champion symbolic and hybrid AI as necessary; some are narrower critiques of specific LLM capabilities; some are programmatic visions for an integrated field. What unites them is a shared concern that the reliability, scalability, and out-of-distribution behavior of pure-LLM systems warrant more empirical scrutiny

than they usually receive. None of these authors, on the reading adopted here, claims that LLMs are devoid of reasoning ability (LLMs can solve many classical reasoning problems, including small blocks-world tasks with appropriate prompting), but the published concerns are about *reliability at scale*, not absolute impossibility, and the present paper adopts that framing throughout.

A separate clarification: it is easy to overstate what symbolic methods guarantee. A SAT solver, an optimal classical planner, or a description-logic reasoner provides *component-level* guarantees: relative to a formal specification, the algorithm is sound and (often) complete. These guarantees do not automatically transfer to the system in which the component is installed. The encoding may be wrong, the axioms ill-chosen, the implementation buggy, the heuristic willing to trade optimality for speed, and the specification itself imperfect. “Precise reasoning” and “formal guarantee” are therefore distinct, and the paper keeps them distinct throughout. Most symbolic systems used in industry (rule engines, business-rule systems, OWL ontologies whose expensive axioms are routinely simplified to keep reasoning tractable) offer no formal guarantees at all. What they offer is *declarative*, *auditable*, and *composable* reasoning, which is a different and often more practically valuable property, [13].

With those clarifications in place, the question addressed here is not whether symbolic methods are obsolete: such a claim would misrepresent the published literature, and it is not a thesis that serious LLM researchers defend in print. The substantive question is which problem characteristics call for symbolic representations, which call for statistical learning, and which call for hybrid architectures. This paper treats the question as a task-and-architecture analysis and is organized around it.

The article is therefore not solely a synthesis of existing results. Alongside the survey, it contributes an original formal layer: Section 7 derives error propagation bounds for search–verification loops, convergence conditions for logic regularization, complexity results for constraint-guided decoding across constraint families, and a unified cost-accuracy model for hybrid architectures, and Section 8 states two characteristic failure modes of hybrid systems as formal propositions. The taxonomic and empirical material supplies the motivation and the parameter estimates for that analysis; the analysis, in turn, converts the survey’s qualitative recommendations into quantitative decision rules.

A note on terminology. By *knowledge-based* (or *symbolic*) methods this paper means systems built on explicit, declarative representations whose inference procedures can be analyzed independently of any

particular dataset: SAT/SMT solvers, constraint programming, classical and optimal planning, inductive logic programming, formal argumentation, answer set programming, and Semantic Web technologies (OWL, SHACL, SPARQL) over knowledge graphs. By *neuro-symbolic* it means any architecture that combines a learned neural component with one or more of these traditions: logic-regularized neural training, constraint-guided decoding, search–verification loops (e.g. LLM-Modulo), and graph-augmented retrieval (e.g. GraphRAG), [10], [11]. Section 3 fixes both terms more carefully.

Contributions. This article provides: (i) an explicit *taxonomy* of knowledge-based methods (Section 3) that distinguishes SAT/SMT, constraint programming, classical and optimal planning, inductive logic programming, formal argumentation, answer set programming, Semantic Web technologies (OWL, SHACL, SPARQL) over knowledge graphs, and neuro-symbolic architectures; (ii) an explicit *survey methodology* (Section 2) including search terms, sources, inclusion criteria, and quantitative counts of the included literature; (iii) a *predictive framework* (Section 4) that links task characteristics to expected architectural fit, accompanied by an explicit mapping (Table 1, Appendix) from characteristics to the symbolic traditions of Section 3; (iv) a *broadened evidence base* (Section 5) across twelve problem classes, with quantitative results assembled in Table 2 (Appendix); (v) a discussion of *integration patterns* (Section 6) emphasizing GraphRAG, knowledge-augmented LLM architectures, logic regularization, constraint-guided decoding, search–verification loops, and probabilistic logic programming; (vi) an original *formal analysis* of the integration patterns (Section 7) that derives error propagation bounds and expected iteration counts for search–verification loops under imperfect verifiers, states convergence conditions for logic-regularized training, establishes complexity results for constraint-guided decoding across constraint families (including an intractability result for arbitrary propositional constraints), and unifies the patterns in a single cost-accuracy trade-off model; (vii) an explicit discussion of *limitations* of both symbolic and hybrid methods (Section 8), including knowledge-acquisition costs, axiom brittleness, the gap between component-level and system-level guarantees, reasoning shortcuts, and the engineering cost of solver-in-the-loop architectures, with the brittleness and shortcut failure modes stated as formal propositions; and (viii) recommendations for benchmark design and practice plus an explicit

future-work agenda (Section 9).

Reading map. The argumentative arc of the paper is as follows. Section 2 establishes the scope of the survey and reports how the evidence was assembled. Section 3 fixes terminology by distinguishing seven symbolic traditions plus the neuro-symbolic combinations of them, which the rest of the paper draws on. Section 4 then articulates nine task characteristics under which symbolic structure is most useful, and Table 1 (Appendix) maps each characteristic to the traditions of Section 3, so that the framework can be used to choose architectures. Section 5 grounds the framework in empirical results across twelve problem classes, summarized in Table 2 (Appendix). Section 6 discusses five patterns for combining neural and symbolic components, plus a unifying view; two of the patterns (LLM-Modulo and GraphRAG) are the dominant industrial configurations today. Section 7 then develops a formal analysis of these patterns: error propagation bounds for search–verification loops, convergence conditions for logic regularization, complexity results for constraint-guided decoding, and a unified cost-accuracy model. Section 8 examines the limitations of both symbolic and hybrid systems, stating two of them as formal propositions. Section 9 draws practical recommendations and a future-work agenda. Section 10 concludes. A glossary of acronyms is provided in Appendix A.

2 Survey Methodology

The empirical core of this paper is a focused review rather than an exhaustive bibliometric mapping of symbolic AI, and this section explicitly delineates its boundaries to facilitate replication and support readers’ assessment of evidence quality. The formal results of Section 7 are original to this article and are not part of the screened literature described below.

Sources. Candidate references were identified from four sources: (i) the proceedings of the major venues that publish symbolic, neuro-symbolic, and reasoning-focused work, namely AAAI, IJCAI, ICML, NeurIPS, ICLR, KR, ECAI, COMMA, ICLP, NeSy, and ISWC; (ii) the corresponding journals, namely *Artificial Intelligence*, *Journal of Artificial Intelligence Research*, *Theory and Practice of Logic Programming*, *ACM Transactions on Information Systems*, *Communications of the ACM*, and *IEEE TPAMI/TNNLS*; (iii) recent surveys on neuro-symbolic computing, knowledge-graph reasoning, GraphRAG, formal argumentation, and answer set programming; and (iv) a small number of pre-prints whose results are not yet published

in conventional venues but are widely cited in the discussion.

Search terms. For each of the twelve problem classes covered in Section 5, the search used combinations of the relevant problem term (e.g., “constraint satisfaction”, “classical planning”, “compositional generalization”, “formal argumentation”, “answer set programming”, “knowledge graph reasoning”, “concept-based interpretability”) with the qualifiers “symbolic”, “neuro-symbolic”, “logic”, “inductive logic programming”, and “large language model”. For integration patterns, the search used “retrieval-augmented generation”, “GraphRAG”, “LLM-modulo”, “constraint-guided decoding”, “logic regularization”, and “probabilistic logic programming”.

Inclusion and exclusion criteria. The review includes works that (a) report quantitative empirical results comparing symbolic, hybrid, and neural approaches on the same benchmark; (b) provide survey-level summaries that themselves cite such empirical comparisons; or (c) define methods that are relevant to the predictive framework even when head-to-head comparisons are not available. It excludes works that report only qualitative claims without empirical support, works restricted to a single proprietary system, and works whose evidence base concerns perception-dominated tasks that lie outside the scope of this paper.

Quantitative summary of the included literature. After screening, the manuscript cites 81 references in total. The temporal distribution illustrates the topic’s currency: a little over half date from 2023 or later, roughly a quarter from 2018–2022, and the remainder from earlier years (with the oldest, the 1960 satisfiability procedure of [14], included because the lineage it founded remains directly relevant). The thematic distribution across the seven traditions of Section 3 is approximately balanced: SAT/SMT and constraint programming, planning and scheduling, inductive logic programming and abstract reasoning, answer set programming, formal argumentation, Semantic Web technologies and knowledge graphs, and interpretability via rule extraction and concept-based methods each receive between four and eight primary references. The remaining references cover integration patterns (GraphRAG, LLM-Modulo, semantic loss, probabilistic logic programming), neuro-symbolic surveys and visions, representative LLM-side positions, foundational or methodological

context, and the optimization and complexity-theory results on which the formal analysis of Section 7 rests. Per problem class in Section 5, the number of supporting citations has a median of 6. It ranges from 1 (for the constraint-satisfaction subsection, which relies on a single Sudoku reference plus an unsurveyed pointer to the operations-research literature) to 8 (for the Semantic Web subsection).

Scope limitations. The twelve problem classes covered are not the entire space of symbolic AI; in particular, work in description-logic reasoning beyond Semantic Web standards, in formal methods for hardware and protocol verification, in probabilistic graphical models, and in symbolic theories of legal and ethical reasoning is mentioned briefly but not surveyed in depth. The choice of classes reflects the areas where empirical head-to-head comparisons with modern LLMs are most readily available. The author acknowledges that this is a narrative review rather than a systematic PRISMA-style review; the quantitative figures above are intended to give the reader a sense of evidence density per topic rather than to substitute for a full systematic protocol. The 2025 IJCAI survey in [12] adopts a similar task-directed organization of neuro-symbolic AI and is a useful complement to the present paper.

3 A Taxonomy of Knowledge-Based Methods

The umbrella term “symbolic AI” conflates several distinct research traditions with different formal foundations, complexity profiles, and engineering practices. This paper therefore makes its usage explicit and distinguishes seven traditions:

1. **Satisfiability and SMT solving.** Decision procedures for propositional and first-order theories, dominated by conflict-driven clause learning (CDCL) for SAT, [15], [16], [17], and by DPLL(T) for SMT, [18], [19]. The output is a yes/no decision or a satisfying assignment; component-level soundness and completeness are central.
2. **Constraint programming and combinatorial optimization.** Global-constraint propagation and branch-and-bound search, often coupled with MaxSAT or mixed-integer programming, [20], [21]. Many production applications pursue and obtain provable optimality on steady-state instances; others, particularly hard online problems, settle for feasibility together with bounded optimality

gaps and runtime predictability, both of which the solver framework makes explicit.

3. **Classical and optimal planning.** Symbolic representations of states, actions, and goals (e.g., PDDL); search guided by domain-independent heuristics or symbolic data structures such as binary decision diagrams, [22], [23].
4. **Inductive Logic Programming (ILP).** Learning relational rules from examples and background knowledge under a logic-program hypothesis space, [24], [25]. The hallmarks are data efficiency, interpretability of the learned rules, and the ability to inject background knowledge directly. Although ILP is sometimes grouped with the broader area of *program synthesis*, which includes DSL-based synthesizers (e.g., FlashFill) and neural program synthesis that need not be expressed in logic clauses, this paper restricts the term to the rule-learning lineage.
5. **Answer Set Programming (ASP).** A declarative paradigm grounded in non-monotonic logic and stable-model semantics, suited to combinatorial search, default reasoning, and preference handling, [26], [27], [28]. ASP forms the symbolic backbone of several neuro-symbolic systems for visual question answering and probabilistic reasoning, [29], [30].
6. **Formal argumentation.** A family of frameworks (abstract argumentation, [31], structured frameworks such as assumption-based argumentation and ASPIC⁺, and bipolar/weighted extensions) that model defeasible reasoning, dialogue, and the dialectical generation of explanations, [32], [33]. Argumentation provides an account of *why* a conclusion holds in the face of conflicting evidence. It underpins a substantial body of work in explainable AI, [34], [35], and conversational systems, [36].
7. **Semantic Web technologies and knowledge graphs.** The W3C standards stack of RDF, RDFS, OWL, [37], SPARQL, [38], and SHACL, [39], provides declarative vocabularies, query languages, and data-validation mechanisms. Knowledge graphs built on these standards now serve as primary symbolic backbones for industrial neuro-symbolic systems, [40], [41], including the GraphRAG family of architectures (Section 6).

These traditions share a commitment to explicit, declarative representations and to inference procedures whose behavior can be analyzed

independently of any particular dataset. They differ substantially in expressiveness, complexity, tooling, and commercial adoption. SAT solvers are widely deployed inside compilers and verification tools, but are largely invisible to end users. Semantic Web technologies, including the description-logic (DL) profiles of OWL (EL, QL, RL) and their dedicated reasoners (HermiT, Pellet, Konclude, ELK), are now standard infrastructure inside large enterprises and the life sciences; this paper treats description-logic reasoning as the DL backbone of the Semantic Web tradition rather than as a separate item, although readers from the KR community may prefer the latter framing. Interactive theorem provers are concentrated in security, cryptography, hardware verification, and the development of certified mathematical libraries, where the consequences of error justify their cost. Recent surveys provide fuller treatments: [10], [11], [42] for neuro-symbolic computing broadly, and [43], [44] for the KG-centric strand.

What “neuro-symbolic” means here.

A second source of confusion is the term “neuro-symbolic”. This paper uses it broadly to refer to any architecture that combines a learned neural component with one or more of the symbolic traditions above. This includes: (a) neural modules that propose candidates and symbolic modules that verify, repair, or score them (LLM-Modulo, VLA_{gent}), [3], [45]; (b) differentiable training objectives that incorporate logical constraints, [46]; (c) neural-probabilistic logic programming combining deep networks with answer set programming, [29]; (d) embedding-based reasoning over knowledge graphs combined with ontological or rule-based inference, [43], [44]; and (e) retrieval-augmented LLMs whose retrieved evidence comes from structured graphs and ontologies (GraphRAG), [47], [48], [49]. In industrial practice, the most common neuro-symbolic configuration today is an LLM coupled with a knowledge graph via GraphRAG-style retrieval, discussed explicitly in Section 6.

4 Predictive Framework: Task Characteristics and Architectural Fit

Not all tasks benefit equally from symbolic, neural, or hybrid approaches. The following task characteristics are the most reliable predictors that knowledge-based or hybrid methods will outperform end-to-end neural baselines. The framework is framed as a guide for choosing architectures, not as a claim of paradigm

superiority.

- **Declarative correctness criteria.** Tasks where success is defined by satisfaction of an explicit specification (satisfiability, formal verification, cost-optimal planning, constraint satisfaction) favor symbolic methods because the specification can be supplied to the solver as input. Component-level guarantees (soundness, completeness, bounded optimality) are central. Examples include CDCL-based SAT solving, [17], and optimal classical planning with expressive cost models, [23].
- **Combinatorial structure and structured search.** Problems with exponential search spaces whose structure admits propagation, conflict learning, or branch-and-bound (CSPs, MaxSAT scheduling, ASP grounding) benefit from decades of solver engineering, [20], [21], [26]. End-to-end neural approaches can solve small instances (current LLMs can solve small blocks-world planning tasks with appropriate prompting), but reliability and scaling remain open problems, [6], [7].
- **Strong priors and explicit domain knowledge.** Domains where humans can articulate hierarchical or relational constraints (taxonomies of objects, physical affordances of agents, or hard structural rules of a system) benefit from declarative encodings, [50]. Neural logic segmentation and human-object interaction detection use first-order constraints to encode object hierarchies and affordances, improving accuracy and zero-shot transfer relative to unconstrained baselines, [51], [52].
- **Data efficiency.** Tasks for which labeled data is scarce, expensive, or distributionally narrow benefit from architectures whose hypothesis space is biased toward few-shot generalization. Inductive logic programming learns relational rules from a handful of examples, [24]; constraint-aware neuro-symbolic training reduces the labeled examples needed to reach a given accuracy on perception-plus-reasoning tasks, [29], [53].
- **Compositional and systematic generalization.** Tasks that require novel combinations of known parts, such as algorithmic generalization to longer inputs, compositional language understanding (SCAN, COGS), and program-induction benchmarks (ARC-AGI), benefit from explicit relational or program-structured representations. Neural Logic Machines generalize from training inputs

of size 20 to inputs of size 100 on algorithmic tasks, [25]; ILP-style systems compose learned predicates over background knowledge.

- **Interpretability and auditability.** Where decisions must be inspected, justified, or contested, symbolic reasoning supplies a natural medium for explanations. Concept-based and logic-explained networks produce first-order-logic explanations directly from neural representations, [54], [55], [56]; argumentation frameworks generate dialectical explanations that have proven useful in explainable AI, [34], [35].
- **Defeasible and contested reasoning.** Tasks in which agents must weigh conflicting evidence or revise conclusions in light of new facts (legal reasoning, decision support, agent dialogue) are well-served by argumentation frameworks and non-monotonic logics, [26], [31], [32].
- **Causal inference and knowledge discovery.** Deriving causal relations from heterogeneous data often demands explicit reasoning. Neural-symbolic architectures that combine knowledge-graph embeddings with rule-based inference can predict causal links absent from the rule base while preserving symbolic interpretability, [57].
- **Grounding in evolving, organization-specific factual content.** Tasks whose correct answer depends on facts that change over time, that are private to an organization, or that must be cited rather than recalled benefit from explicit, queryable knowledge bases. Retrieval from a symbolic knowledge graph at inference time both reduces hallucination and supplies citable evidence, [47], [48], [49], [58], [59].

Tasks dominated by unstructured perception, fluent generation, or broad common-sense association remain the natural domain of large neural models. Even there, symbolic guidance via retrieval, constraint-guided decoding, or post hoc verification can improve robustness and auditability.

Mapping characteristics to traditions. Table 1 (Appendix) makes the connection between the nine characteristics above and the seven symbolic traditions of Section 3 explicit, so that the predictive framework can be used to select an architecture in practice. A given task may exhibit several characteristics and may therefore call for more than one tradition; the mapping is intended as a lookup rather than a partition.

5 Empirical Evidence

This section compiles experimental results from recent literature showing that knowledge-based and neuro-symbolic methods retain measurable advantages over strong neural and LLM baselines. The aim is calibration rather than advocacy: throughout the section, the text reports what specific systems achieve on specific benchmarks rather than asserting paradigm-level superiority. Each subsection notes, after its empirical content, which task characteristics from Section 4 the problem class primarily exemplifies, thereby closing the loop with the predictive framework. Table 2 (Appendix) presents a structured summary that includes the system, baseline, dataset, metric, and observed result.

5.1 Satisfiability and SMT

The Boolean satisfiability problem is prototypical of NP-complete tasks. Modern conflict-driven clause-learning (CDCL) solvers such as MiniSAT, Glucose, and Kissat routinely handle industrial benchmarks with millions of variables and serve as primary tools for verification, planning, and combinatorial design, [14], [15], [16], [17]. SMT solvers extend this success to first-order theories, [18], [19]. Neural and neuro-augmented SAT methods, such as NeuroSAT and its successors, have produced interesting results on small instances but do not match CDCL solvers at the industrial scale. The reason is concrete and technical: CDCL builds and learns from the *implication graph* of a partial assignment, deriving conflict clauses that prune large regions of the search space and that compactly summarize unsatisfiable cores. Current neural SAT models lack an analogous mechanism to represent and reason over this graph structure during solving. Recent neuro-symbolic work explores ways to learn conflict-clause-like representations and to guide CDCL search with neural heuristics. Still, these directions have not yet displaced standard CDCL on industrial benchmarks. *Characteristics primarily exemplified: declarative correctness criteria, combinatorial structure.*

5.2 Constraint Satisfaction

Sudoku puzzles illustrate, in a deliberately small setting, how constraint programming enforces row, column, and subgrid all-different constraints to reach complete solutions efficiently. A hybrid alldifferent-Tabu search algorithm reaches 100% accuracy on 9×9 Sudoku puzzles with faster runtimes than approximate neural heuristics, [20]. The same constraint-propagation machinery scales: industrial CP and MIP solvers built on global constraints and branch-and-bound are the dominant production

tools for configuration, vehicle routing, scheduling, and timetabling, with empirical performance on real-world instance distributions that neural baselines have yet to match. This paper does not survey the large operations-research literature in detail here; the example matters because it shows that the structural argument for symbolic methods on combinatorial tasks generalizes far beyond the puzzle setting. *Characteristics primarily exemplified: combinatorial structure, declarative correctness criteria.*

5.3 Abstract Reasoning and Program Induction

The Abstraction and Reasoning Corpus (ARC), proposed as a measure of fluid intelligence in [60], constituted a consistent negative result for both symbolic and neural systems for several years. The situation has changed markedly in recent years. The 2024 ARC Prize lifted public state-of-the-art on the ARC-AGI-1 evaluation set from roughly 33% to 55.5%, driven by deep-learning-guided program synthesis and test-time training, and in December 2024, OpenAI’s o3 reached 87.5% on the semi-private ARC-AGI-1 set, at an estimated cost of roughly \$4,500 per task at maximum compute, approximately $172\times$ the low-compute configuration; these cost figures are vendor-disclosed rather than peer-reviewed and should be read as such, [61]. The ARC community responded by releasing ARC-AGI-2, on which leading reasoning models scored in the low single digits while humans remained near ceiling at the time of the ARC Prize 2024 technical report; the 2025 ARC Prize cycle remains ongoing at the time of writing, so the gap may be narrower by the time this paper is in print. Two lessons follow: scaled test-time search now solves a substantial fraction of the original benchmark, but performance does not transfer cleanly to a redesigned, harder version, which is precisely what an out-of-distribution test should reveal.

In parallel, two neuro-symbolic systems remain useful reference points for the broader class of algorithmic generalization tasks. Neural Logic Machines learn relational rules and generalize from small training instances to substantially larger inputs, for example from family-tree problems with 20 entities to ones with 100, and from short sorting arrays to longer ones, [25]. SLASH integrates neural networks with answer set programming and reports strong results on visual question answering and arithmetic, where the symbolic backbone supplies inductive bias, [29]. These results suggest that the inductive bias supplied by an explicit symbolic component remains useful even as scaled neural systems improve on the headline benchmark.

Characteristics primarily exemplified: compositional and systematic generalization, data efficiency.

5.4 Formal Mathematics and Theorem Proving

Interactive proof assistants (Coq, Lean, Isabelle, HOL) produce explicit proof objects that can be machine-checked, which is the operative form of correctness in formal mathematics. They are also used widely in academic mathematics and formal-methods teaching. Their use in the wider economy is concentrated in domains where the cost of an undetected error is severe: hardware design, cryptographic protocols, safety-critical control software, and the development of certified mathematical libraries. The point to underline is narrow: where verifiability is mandatory, interactive provers remain the standard tool. A particularly informative recent neuro-symbolic result is AlphaGeometry, which combines a language model that proposes auxiliary constructions with a symbolic deductive engine (DD+AR) and, on the IMO-AG-30 benchmark of 30 IMO-style geometry problems, solves 25/30 within the standard competition time limit, compared with the 10/30 reported in [62] for Wu’s method as a fully symbolic baseline. Subsequent work has shown, however, that Wu’s method is stronger than the reported baseline once it is implemented carefully: Wu’s method alone solves 15/30, and a Wu + DD+AR ensemble solves 21/30, four problems short of AlphaGeometry rather than fifteen, [63]. The corrected comparison remains favorable to the neuro-symbolic configuration; nevertheless, the margin between the strongest fully symbolic baseline and the LLM-augmented system is substantially smaller than the original report suggested. A successor system, AlphaGeometry 2, solves all 30 problems in IMO-AG-30 and 42/50 on the broader IMO-AG-50 benchmark introduced alongside it by enlarging the symbolic engine’s vocabulary, broadening the language model’s training distribution, and adding a knowledge-sharing search mechanism, [64]. The trajectory is confirmed; the underlying generator-plus-verifier architecture is unchanged. Within this benchmark and others where solutions can be checked formally, the most effective use of LLMs is as a generator coupled to a verifier rather than as a stand-alone reasoner, [3], [13]. *Characteristics primarily exemplified: declarative correctness criteria, interpretability, and auditability (proof objects).*

5.5 Planning and Scheduling

Classical symbolic planners guarantee completeness and, with admissible heuristics, optimality with

respect to a stated cost model. Recent work extends symbolic search to handle conditional effects and state-dependent costs, with reported reductions in search time and memory relative to prior optimal planners on standard IPC domains, [23]. In scheduling, MaxSAT formulations of the business-to-business meeting scheduling problem outperform both constraint programming and mixed-integer programming on the reported instance distributions; on the hardest reported instances, the MaxSAT formulation solves to optimality while the CP and MILP baselines time out, with order-of-magnitude differences in median runtime on the moderate-difficulty instances that both methods complete, [21]. Systematic evaluations of large reasoning models confirm that LLM-based planners frequently produce invalid or incomplete plans, and that reliable planning quality emerges only when LLMs are coupled with classical verifiers in LLM-Modulo configurations, [3], [7], [8], [65]. Current LLMs can, of course, solve small instances of benchmark planning problems, such as Blocks World, when prompted appropriately; the published concerns are about reliability at scale and on out-of-distribution instances, not about absolute impossibility. *Characteristics primarily exemplified: declarative correctness criteria, combinatorial structure, strong priors.*

5.6 Compositional Generalization

Vanilla sequence-to-sequence models have historically failed to generalize compositionally on benchmarks such as SCAN and COGS, motivating an active sub-literature on inductive biases for systematic generalization. Modern LLMs perform better than older sequence-to-sequence models on many SCAN/COGS splits, particularly with chain-of-thought prompting and few-shot exemplars. Yet systematic failures persist in length, productivity, and substitutivity splits, and the evidence falls well short of establishing that compositional generalization is solved, [6], [7]. A separate line of work argues that symbolic structure is the more reliable route to systematic generalization. Inductive logic programming learns relational rules from a handful of examples, [24], and Neural Logic Machines demonstrate that an architecture organized around lifted rules can generalize from small training instances to substantially larger inputs on algorithmic tasks (sorting, shortest paths, family-tree relations) where standard transformers struggle, [25]. This pattern is consistent with the broader theme of this survey: symbolic structure provides a strong inductive bias when systematic generalization is the property under test, whereas LLMs provide broad coverage where the test distribution overlaps with

what was learned during pre-training. *Characteristics primarily exemplified: compositional and systematic generalization.*

5.7 Visual Reasoning

Neuro-symbolic methods have made measurable contributions to visual reasoning. LogicSeg encodes hierarchical coherence over spriors/domainies as first-order logic constraints and reports gains in mean IoU of roughly one to two percentage points on ADE20K and Cityscapes relative to strong transformer-based segmentation baselines (SegFormer, Mask2Former), a modest but consistent improvement obtained by adding a logical constraint term to the loss rather than changing the segmentation architecture, [51]. Neural-Logic Human-Object Interaction detection encodes affordances and proxemics as logical constraints on HICO-DET and improves detection mAP while supporting zero-shot generalization to unseen interaction categories, [52]. The modular NSGRAPH architecture combines optical graph recognition, OCR, and answer set programming on the CLEGR visual-graph QA dataset, achieving an overall average accuracy of 73%, demonstrating the feasibility of ASP-backed neuro-symbolic VQA, [30]. CLEGR uses controlled synthetic scenes of graph diagrams, so it is a strong test of the symbolic-reasoning side of VQA but not of perception under naturalistic distribution shift; broader vision-language benchmarks such as GQA-OOD, CLEVRER, and NLVR2 have not been evaluated for this method. The VLAgent architecture couples an LLM plan generator with neural modules and a symbolic verifier on compositional visual reasoning tasks, reporting accuracy gains over end-to-end vision-language baselines while exposing intermediate reasoning steps that humans can inspect, [45]. *Characteristics primarily exemplified: strong priors / domain knowledge, interpretability.*

5.8 Causal Inference

Neuro-symbolic architectures combine knowledge-graph embeddings with rule-based reasoning to derive causal knowledge. In a smart-grid case study, a neural-symbolic system predicts causal links absent from the symbolic rule base and improves Hits@5 over purely symbolic and purely embedding baselines, [57]. The empirical base for neuro-symbolic causal discovery is still narrower than for satisfiability or planning, but two adjacent literatures merit attention. Probabilistic logic programming over structural causal models provides a declarative substrate for counterfactual reasoning that hybrid systems can call into, and recent work couples LLMs with classical causal-discovery

algorithms by using the LLM to propose plausible variable orderings or pairwise causal hypotheses that a symbolic search then verifies. A comprehensive comparison across these lines would also need to engage instrumental-variable methods and structural causal models properly; this gap is flagged explicitly in the future-work agenda. *Characteristics primarily exemplified: causal inference and knowledge discovery, interpretability.*

5.9 Formal Argumentation and Explainable AI

Formal argumentation theory is one of the most active branches of contemporary symbolic AI, with sustained output in the proceedings of IJCAI, AAAI, KR, ECAI, and COMMA, as well as in the journals *Artificial Intelligence* and *Argument and Computation*. Building on the seminal abstract argumentation framework of [31], the field has developed structured frameworks (ASPIC⁺, assumption-based argumentation, ABA⁺), bipolar and weighted extensions, and a rich complexity-theoretic understanding of acceptance problems, [32], [33]. Argumentation has become a primary symbolic substrate for explainable AI: dialectical explanations naturally surface supporting and attacking reasons, model defeasible inference under conflicting evidence, and support contestability, [34], [35]. Recent work integrates argumentation with conversational agents, producing chatbot architectures whose responses can be traced back to argument structures rather than to opaque generation, [36]. Argumentation also intersects with answer set programming through claim-centric argumentation and ABA-to-AF translations, [33]. From the standpoint of the present survey, argumentation is significant precisely because it captures a property that statistical learning does not naturally provide: a representation of *why* a conclusion holds, expressed in terms that humans can inspect and challenge. *Characteristics primarily exemplified: defeasible/contested reasoning, interpretability, and auditability.*

5.10 Answer Set Programming

Answer Set Programming (ASP) is a declarative programming paradigm grounded in non-monotonic logic and stable-model semantics, [26], [27], well suited to combinatorial problems, defaults, and preference handling, with mature solvers (clingo, DLV) and an active literature on explanations of answer sets, [28]. Several neuro-symbolic ASP systems embed neural atoms into ASP programs, enabling joint training of perception and reasoning. NeurASP introduced the basic technique: treat

each neural network output as a probability distribution over atomic facts, then constrain training through ASP rules, [53]. On MNIST-Addition and Sudoku, NeurASP achieves accuracy comparable to DeepProbLog with notably reduced per-iteration training time, attributed to Clingo's enumeration of stable models being faster than DeepProbLog's per-instance construction of a sentential decision diagram; constraint-aware training also allows neural perception modules to learn from rules rather than only from labels. SLASH extends this approach to a more general probabilistic semantics and improves both accuracy and the stability of training across runs, [29]; the NSGRAPH system pairs ASP-backed reasoning with optical graph recognition and OCR on the CLEGR visual-graph QA benchmark, [30]. ASP exemplifies how a mature declarative paradigm can serve as the symbolic side of a neuro-symbolic architecture without sacrificing transparency. The broader family of probabilistic logic programming, including ProbLog and DeepProbLog, is covered as an integration pattern in Section 6. *Characteristics primarily exemplified: combinatorial structure, defeasible reasoning (via stable-model semantics), data efficiency.*

5.11 Knowledge Graphs, OWL/SHACL/SPARQL, and Reasoning

The Semantic Web standards stack (RDF, RDFS, OWL, [37], SPARQL, [38], and SHACL, [39]) provides declarative vocabularies, query languages, and data-validation mechanisms that are widely adopted in industry, the life sciences, and government data infrastructure, [40], [66]. OWL-based knowledge graphs support ontological reasoning ranging from class hierarchies and consistency checking to more expressive description-logic inference. The tractable OWL EL, QL, and RL profiles admit polynomial-time reasoners that scale to large biomedical ontologies: ELK reasons over the full SNOMED CT terminology (over 3×10^5 classes) in seconds, and dedicated reasoners such as HermiT, Pellet, and Konclude handle expressive OWL DL on ontologies up to several million axioms in production deployments. SHACL provides standardized constraints over RDF graphs and supports automated validation reports that surface specific violations against an explicit shape graph, [39], [67]. In large deployments, axioms that slow reasoning at high triple counts are routinely simplified, trading expressive completeness for operational tractability without abandoning the underlying schema. Recent work argues that OWL-based KGs remain an underused substrate for neuro-symbolic systems and

identifies concrete integration opportunities, [41], [43], [44]. The connection to LLMs is direct: GraphRAG architectures use KGs as the symbolic store from which evidence is retrieved and supplied to an LLM at inference time (Section 6). *Characteristics primarily exemplified: grounding in evolving, organization-specific factual content, interpretability, and auditability.*

5.12 Interpretability via Concept-Based and Rule-Extraction Methods

A complementary line of work exploits symbolic structures to render neural decisions auditable. It is a direct response to the broader interpretability and accountability concerns now associated with deployed AI systems. Rule-extraction techniques distill neural classifiers into rule sets that approximate the network’s behavior, [68], [69], [70]. Concept-based architectures introduce an explicit concept layer between perception and decision: concept bottleneck models predict named concepts that humans can inspect and intervene on, [56]; logic-explained networks supply propositional Boolean explanations over concept activations using an entropy-based criterion, [55]; deep concept reasoners (DCR) build differentiable rule structures over concept embeddings and, on the benchmarks reported in [54], recover ground-truth rules while achieving accuracy comparable to opaque concept-based baselines and clearly above earlier rule-extraction methods (the paper reports gains of several accuracy points on relational tabular benchmarks while producing the same per-class rule structures used as ground truth), [54]; neuro-symbolic class-expression learners exploit description-logic structure for interpretable learning on ontological data, [71]. These methods illustrate the practical contribution of symbolic structure to current neural systems: explanations that are simultaneously faithful to the model and intelligible to a human auditor, achieved without the accuracy collapse that earlier rule-extraction methods sometimes incurred. *Characteristic primarily exemplified: interpretability and auditability.*

6 Integration Patterns

Hybrid architectures exploit the complementary strengths of neural and symbolic components. This section discusses five concrete integration patterns prominent in current practice (logic regularization, constraint-guided decoding, search–verification loops, graph retrieval, and probabilistic logic programming), followed by a unifying perspective on what they have in common.

Logic regularization and constraint penalties.

In supervised learning, symbolic knowledge can be injected through differentiable regularizers. Given a task loss $\mathcal{L}_{\text{task}}$ and a set of logical constraints Φ , one can define a differentiable logic loss $\mathcal{L}_{\text{logic}}(\theta, \Phi)$ and train with

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{task}}(\theta) + \lambda \mathcal{L}_{\text{logic}}(\theta, \Phi), \quad (1)$$

where λ trades off empirical risk against constraint satisfaction. Operationalizing $\mathcal{L}_{\text{logic}}$ requires a differentiable relaxation of Boolean satisfaction; common choices include continuous t-norm fuzzy logics (product, Gödel, Łukasiewicz) and the *semantic loss*, which equals the negative log probability that a model’s predictions, viewed as a distribution over assignments, satisfy a propositional constraint Φ , [72]. Recent work derives a sample-based logical regularizer (L-Reg) that bridges logical analysis with image classification, improving generalization across multi-domain image classification and generalized category discovery while also producing interpretable reasoning, [46]. Logic regularization is one of the clearest illustrations of how symbolic knowledge serves as an inductive bias for data efficiency and robustness.

Constraint-guided decoding. During inference, symbolic constraints can prune or guide the search space. The intended objective is to penalize violations of declarative constraints Φ in the generated output:

$$\hat{y} \approx \arg \max_y [\log p(y | x) - \gamma C(y, \Phi)], \quad (2)$$

where $C(y, \Phi)$ counts violated constraints and γ controls the penalty. The arg-max over full sequences is intractable for LLMs, so the equation expresses intent rather than an algorithm. In practice, the constraint is enforced incrementally at each decoding step, for example, by zeroing out the probabilities of tokens that would violate the grammar, type signature, schema, or trie under consideration. The decoding-time computation is tractable for context-free, regular, and trie-based constraint families, and the pattern is now common in structured generation, code synthesis, and tool use.

Search–verification loops (LLM-Modulo).

In this pattern, a neural model proposes candidates that a symbolic verifier then checks. Two variants coexist: in a *filter* variant, candidates that fail verification are rejected, and the model is queried again; in a *feedback* variant, the verifier’s diagnostic output (failing constraint, type error, planner-reported precondition violation, counter-example trace) is fed back to the LLM as additional context for the

next attempt. The general pattern is formalized as the LLM-Modulo framework in [3], where it is argued to be the appropriate architectural choice when planning quality matters. The pattern generalizes: LLMs propose code that compilers and unit tests check; LLMs propose proofs that proof assistants verify; LLMs propose plans that classical planners validate, [7], [45], [62]. The configuration explicitly separates generative breadth from verifiable correctness and is now the prevailing pattern in industrial reasoning systems.

Graph Retrieval-Augmented Generation (GraphRAG) and knowledge-augmented LLMs. A fourth pattern, now arguably the most widely deployed neuro-symbolic configuration in industry, couples an LLM with a knowledge graph through retrieval. Standard retrieval-augmented generation retrieves text chunks, [58]; GraphRAG retrieves graph elements (nodes, triples, paths, subgraphs) from a pre-constructed knowledge graph, exploiting the relational structure to support multi-hop reasoning and reduce hallucinations relative to either standalone LLMs or flat text-based RAG. Three contemporary references survey the area from different angles: a workflow-centric taxonomy is developed in [47], a domain-oriented one in [48], and [49] describes Microsoft’s open-source GraphRAG system, which is the most widely deployed concrete implementation. Think-on-Graph and related architectures iteratively traverse a KG under LLM guidance, producing reasoning traces grounded in the underlying graph that humans can audit, [59]. The configuration combines the linguistic capabilities of the LLM with the declarative, queryable, and auditable nature of the underlying graph, and it is increasingly supported by all major triplestore vendors and by the W3C standards stack. Among knowledge-augmented architectures, GraphRAG is the one most directly addressed to the practical concerns that motivate this paper: LLM hallucination, lack of organization-specific knowledge, and the need for citable evidence.

Probabilistic logic programming as an integration substrate. A complementary substrate is probabilistic logic programming (PLP), which combines logic programming with probabilistic semantics. ProbLog attaches probabilities to logical clauses and provides exact and approximate inference over the resulting distribution semantics, [73]; DeepProbLog extends this by letting probabilistic atoms be evaluated by neural networks, so that perception and structured inference are trained end-to-end while preserving the declarative reading of the logic side, [74]. The same idea

underlies NeurASP and SLASH, which integrate neural networks with answer set programming (Section 5.10) and report state-of-the-art results on visual question answering and arithmetic, where the symbolic backbone supplies inductive bias, [29], [53]. Unlike logic regularization, where the symbolic constraint enters as a penalty on a separate neural model, and unlike search–verification loops, where the solver acts after a candidate is produced, PLP fuses the two: probabilistic inference *over the logic program* propagates gradients through the neural predicates that supply its atomic facts, so symbolic and neural components are part of a single inference procedure.

Symbolic knowledge as inductive bias (a unifying view). A common thread runs through these patterns. In each case, symbolic structure functions as an *inductive bias* that improves data efficiency, robustness, and compositional generalization of neural models. This is the central thesis of the third-wave neuro-symbolic agenda, [10], and of the neuro-symbolic AI tutorial overview in [11]. It clarifies what symbolic AI brings to the modern statistical paradigm: not paradigmatic competition, but a principled way of supplying structure (constraints, schemas, ontologies, proof obligations) that neural training cannot easily induce from data alone.

7 A Formal Analysis of Integration Patterns

The integration patterns of Section 6 are usually presented architecturally, as block diagrams connecting a neural proposer to a symbolic checker, regularizer, or store. This section develops a complementary formal treatment. It derives error propagation bounds and expected iteration counts for search–verification loops under imperfect verifiers, states convergence conditions for logic-regularized training, establishes complexity results for constraint-guided decoding across constraint families, and unifies these patterns into a single cost– accuracy trade-off model. The assumptions are deliberately minimal, and the proofs are elementary; the value of the statements lies in making explicit which properties of the components (verifier soundness, constraint family, smoothness constants) control system-level behavior. Propositions 6 and 7 in Section 8 extend the same formal treatment to two characteristic failure modes of hybrid systems.

7.1 Error Propagation in Search–Verification Loops

Definition 1 (Search–verification loop). A search–verification loop consists of a generator G that emits candidate solutions and a verifier V that accepts or rejects each candidate. In each round, G produces one candidate, V checks it, and the loop returns the first accepted candidate; after a budget of k rounds without acceptance, the loop reports failure. The generator is modeled by the probability $p \in (0, 1]$ that an emitted candidate is correct, with rounds independent and identically distributed; the verifier is modeled by its false-acceptance rate $\alpha = \Pr[V \text{ accepts} \mid \text{candidate incorrect}]$ and false-rejection rate $\beta = \Pr[V \text{ rejects} \mid \text{candidate correct}]$.

Proposition 1 (Error propagation and cost of the loop). Let $q = p(1 - \beta) + (1 - p)\alpha$ denote the per-round acceptance probability, and assume $q > 0$. Then: (i) the loop terminates with an accepted candidate within k rounds with probability $1 - (1 - q)^k$; (ii) conditional on returning a candidate, the probability that the returned candidate is incorrect is

$$\varepsilon_{\text{sys}} = \frac{(1 - p)\alpha}{p(1 - \beta) + (1 - p)\alpha}, \quad (3)$$

independent of the budget k ; and (iii) with an unbounded budget, the expected number of generator and verifier calls before acceptance is $1/q$ each.

Proof. Rounds are independent, and in each round the disjoint events “accept a correct candidate” and “accept an incorrect candidate” occur with probabilities $p(1 - \beta)$ and $(1 - p)\alpha$ respectively, so acceptance occurs with probability q and the number of rounds until first acceptance is geometric with parameter q , giving (i) and (iii). For (ii), the type of the first accepted candidate is determined by which of the two disjoint acceptance events occurs first; by the memorylessness of the process, its conditional distribution is proportional to the per-round probabilities, yielding Equation (3). $\square$

Corollary 1 (Soundness transfer). If the verifier is sound ($\alpha = 0$) then $\varepsilon_{\text{sys}} = 0$ for every generator quality $p > 0$ and every completeness deficit $\beta < 1$: system-level correctness is controlled entirely by verifier soundness, while generator quality and verifier completeness affect only the expected cost $1/(p(1 - \beta))$. For small α , the system error degrades gracefully, with $\varepsilon_{\text{sys}} \leq \alpha(1 - p)/(p(1 - \beta))$ growing linearly in α .

Corollary 1 is the formal content of the LLM-Modulo position, [3]: coupling an unreliable

generator to a sound verifier yields a reliable system whose reliability does not depend on the generator, and the generator’s competence determines cost rather than correctness. In the feedback variant, where the verifier’s diagnostic output is returned to the generator, the per-round success probabilities $p_1 \leq p_2 \leq \dots$ are nondecreasing when feedback is informative, and the termination probability within k rounds improves accordingly to $1 - \prod_{t=1}^k (1 - q_t)$ with $q_t = p_t(1 - \beta) + (1 - p_t)\alpha$; the error expression (3) then holds round by round with p_t in place of p .

7.2 Convergence Conditions for Logic Regularization

Consider the regularized objective of Equation (1), $\mathcal{L}(\theta) = \mathcal{L}_{\text{task}}(\theta) + \lambda \mathcal{L}_{\text{logic}}(\theta, \Phi)$.

Assumption 1. $\mathcal{L}_{\text{task}}$ and $\mathcal{L}_{\text{logic}}$ are differentiable with L_1 - and L_2 -Lipschitz gradients, respectively; both are bounded below, and stochastic gradients are unbiased with variance bounded by σ^2 .

Proposition 2 (Convergence of logic-regularized training). Under Assumption 1, $\mathcal{L}$ has L_λ -Lipschitz gradient with $L_\lambda = L_1 + \lambda L_2$, and stochastic gradient descent with step size $\eta \leq 1/L_\lambda$ run for T iterations satisfies $\min_{t \leq T} \mathbb{E} \|\nabla \mathcal{L}(\theta_t)\|^2 = O(\sqrt{L_\lambda} \sigma / \sqrt{T})$, the standard nonconvex rate, [75]. The admissible step size therefore shrinks as $1/\lambda$, so the constraint weight trades constraint enforcement against optimization difficulty.

Proposition 3 (Penalty consistency). Suppose in addition that the feasible set $F = \{\theta : \mathcal{L}_{\text{logic}}(\theta, \Phi) = 0\}$ is nonempty, that $\mathcal{L}_{\text{task}}$ and $\mathcal{L}_{\text{logic}}$ are continuous, and that the sublevel sets of $\mathcal{L}$ are compact. Then, as $\lambda \rightarrow \infty$, every limit point of global minimizers θ_λ^* of $\mathcal{L}$ is a global minimizer of $\mathcal{L}_{\text{task}}$ restricted to F . This is the classical penalty-method result, [76], applied to the logic loss.

Proposition 3 identifies the property a differentiable relaxation must have for regularization to enforce the constraints in the limit: $\mathcal{L}_{\text{logic}}(\theta, \Phi) = 0$ must hold exactly when the constraints are satisfied. The semantic loss satisfies this by construction, vanishing precisely when the model places all probability mass on satisfying assignments, [72], and the product, Gödel, and Łukasiewicz t-norm relaxations satisfy it on the Boolean vertices of the relaxation. Together, Propositions 2 and 3 give concrete convergence conditions for the pattern: smoothness of both losses, a relaxation that is exact at zero, and a constraint weight chosen with the $1/\lambda$ step-size scaling in mind.

7.3 Complexity of Constraint-Guided Decoding

Constraint-guided decoding (Equation (2)) is implemented by masking, at each of n decoding steps, the tokens of the vocabulary Σ whose selection would make the constraint unsatisfiable. The relevant computational problem is *extensibility*: given a prefix and a constraint Φ , decide whether the prefix can be completed to an output satisfying Φ .

Proposition 4 (Complexity by constraint family).

(i) If Φ is a regular language recognized by a deterministic finite automaton with state set Q from which dead states have been removed, extensibility is decidable in $O(1)$ time per token after $O(|Q||\Sigma|)$ preprocessing, so constraint-guided decoding costs $O(n|\Sigma|)$ overall; lexicon and trie constraints are the special case of an acyclic automaton. (ii) If Φ is a context-free language, incremental Earley recognition decides extensibility in $O(n^2)$ time per step and $O(n^3)$ overall in the general case, and in linear total time for deterministic, LR-parsable grammars, which cover the programming-language and schema grammars used in structured generation. (iii) If Φ is an arbitrary propositional formula over the output variables, extensibility is NP-complete: with the empty prefix, it is exactly propositional satisfiability, [77]. Consequently, no polynomial-time exact masking procedure exists for arbitrary propositional constraints unless $P = NP$.

Proof. (i) The decoder maintains the current automaton state; a token is admissible exactly when its transition leads to a live state, which is a table lookup after dead-state elimination. (ii) These are standard incremental-parsing bounds. (iii) Hardness follows by the trivial reduction from satisfiability; membership in NP holds because a completing suffix is a polynomial-size certificate. $\square$

Proposition 4 explains an otherwise empirical regularity of Section 6: deployed constrained decoders restrict attention to regular, trie-based, and deterministic context-free families, because those are precisely the families for which exact masking is tractable at generation time. It also positions knowledge compilation, in which Φ is compiled offline into a tractable circuit as in the semantic-loss line of work, [72], as the systematic way to move the intractability of case (iii) from decoding time to a one-time compilation step.

7.4 A Unified Cost-Accuracy Model

The patterns can be compared within one model. Let c_G denote the cost of one generator call, c_S the cost of one symbolic call (verification, masking, or inference), $A(H) \in [0, 1]$ the task accuracy of system

H , and $\mathbb{E}[C(H)]$ its expected cost. The patterns instantiate as follows. Pure generation has $\mathbb{E}[C] = c_G$ and $A = p$. The search-verification loop with a sound verifier has $A = 1$ conditional on termination and, by Proposition 1, $\mathbb{E}[C] = (c_G + c_S)/(p(1 - \beta))$. Constraint-guided decoding has $\mathbb{E}[C] = c_G + nc_S$ with the per-step masking cost c_S given by Proposition 4, and guarantees satisfaction of Φ by construction while leaving semantic accuracy to the model. Logic regularization moves the symbolic cost entirely into training, leaving inference cost at c_G , with the accuracy effect governed by Propositions 2 and 3. Probabilistic logic programming incurs c_S inside every training and inference step, where exact inference is #P-hard in general and tractable on compiled circuits, [73], [74].

Proposition 5 (When verification pays). *Let $U > 0$ be the utility loss incurred by returning an incorrect answer, and compare pure generation against a search-verification loop with a sound verifier under the model of Proposition 1. The loop yields higher expected utility exactly when*

$$(1 - p)U > \frac{c_G + c_S}{p(1 - \beta)} - c_G. \quad (4)$$

In particular, verification is favored when errors are costly and moderately frequent, and it loses its advantage in both limits: as $p \rightarrow 1$ the left side vanishes, and as $p \rightarrow 0$ the right side diverges because the loop must cycle many candidates before one passes.

Proof. Pure generation has expected utility loss $(1 - p)U$ plus cost c_G ; the loop has error zero and expected cost $(c_G + c_S)/(p(1 - \beta))$ by Proposition 1(iii) and Corollary 1. Comparing the two expected totals gives Equation (4). $\square$

Inequality (4) is a decision rule for practitioners: it makes precise the observation of Section 8 that solver-in-the-loop overhead can dominate a system, and it quantifies precisely when that overhead is justified. The analysis in this section rests on simplifying assumptions, most notably the independence of rounds in Proposition 1, which the feedback variant deliberately violates in a favorable direction, and the reduction of generator behavior to a single scalar p . The statements should therefore be read as first-order models whose purpose is to expose which component properties drive system behavior; extending them to correlated candidate draws, cost-aware iteration schedulers, and PAC-style joint bounds over learner and solver is part of the future-work agenda of Section 9.2.

8 Limitations and Open Challenges

A fair appraisal of knowledge-based methods and of the hybrid systems built on them must surface their limitations. The discussion is organized under five headings.

Knowledge acquisition is expensive.

Constructing high-quality rule bases, ontologies, and knowledge graphs requires sustained human effort and domain expertise. In large industrial deployments, axioms are often simplified to keep reasoning tractable at scale, which reduces the formal value of the underlying logic without entirely removing its benefits. Several research lines mitigate but do not eliminate this cost: inductive logic programming and class-expression learning automate part of the rule-discovery process from examples, [24], [71]; knowledge-graph completion fills in missing triples, [43], [44]; LLM-assisted ontology curation extracts candidate axioms from text that an expert then verifies; and SHACL-style validation supplies feedback signals on machine-extracted graphs, [39], [67]. The bottleneck is best treated as a workflow problem, combining these tools with human-in-the-loop validation, rather than as a single technical question awaiting a single solution.

Component-level guarantees are not system-level guarantees.

A SAT solver, a classical planner, or a description-logic reasoner is sound and complete relative to its formal specification. The system in which it is embedded usually is not. The encoding may be wrong, the axioms may be misspecified, the heuristic may be willing to trade optimality for speed, and the implementation may be buggy. “Precise reasoning” and “formally guaranteed correctness” are distinct properties, and confusing them inflates symbolic methods’ practical claims, [78]. Modern LLMs and modern symbolic systems both routinely fail to deliver formal system-level correctness, but for opposite reasons: LLMs because their inference procedure is statistical and offers no formal guarantees in the first place; symbolic solvers because the encoding, axioms, and surrounding code that connect a sound solver to a real-world task are themselves not formally verified. The same pattern recurs in evaluations of hybrid systems: AlphaGeometry’s initially reported 25/30 was substantially reframed once Wu’s method was implemented carefully in [63], and LLM-Modulo evaluations on PlanBench consistently show that end-to-end task accuracy hides component-level errors that only a verifier-in-the-loop exposes, [3],

[7]. The methodological lesson is general, and applies to any complex AI system: report component-level and system-level evaluations separately.

NP-hardness imposes intrinsic limits. Many of the problems for which symbolic methods are most effective are NP-hard. For sufficiently large instances, optimality is intractable, and what solvers actually deliver is feasibility, bounded optimality gaps, runtime predictability, and structured backtracking. The framing matters: combinatorial optimization does not demand absolute correctness guarantees; it benefits from bounded optimality, structured search, and post hoc verifiability.

Brittleness to misspecified or shifting axioms.

Symbolic systems are sensitive to the axioms they are given. Misspecification produces incorrect inferences silently. The sensitivity can be stated formally.

Proposition 6 (Brittleness of classical axiom sets). *Let KB be a consistent set of axioms in classical propositional or first-order logic and let Cn denote classical consequence. (i) There exists a single axiom a such that $Cn(KB \cup \{a\})$ is the set of all formulas: any a inconsistent with KB suffices so that a one-axiom misspecification can corrupt every conclusion of the system. (ii) For any formula φ with $\varphi \notin Cn(KB)$ and $\neg\varphi \notin Cn(KB)$, a single added axiom decides φ in either direction. Consequently, the map $KB \mapsto Cn(KB)$ admits no stability guarantee under axiom edits: the change in the conclusion set is not bounded by any function of the number of edited axioms.*

Proof. (i) is ex falso quodlibet; (ii) follows by adding φ or $\neg\varphi$ as the new axiom. $\square$

Proposition 6 is elementary, but it makes precise why knowledge-base maintenance is not optional and why formalisms with non-classical semantics matter in practice: paraconsistent logics, stable-model semantics in ASP, and argumentation semantics localize the damage of a faulty rule to the models or extensions in which that rule participates, rather than propagating it globally. No semantics, however, can recover conclusions whose supporting axioms are wrong; shifting domains therefore require maintenance of the symbolic substrate, which is itself an ongoing engineering cost. Neuro-symbolic architectures that learn rules from data (ILP in particular, and neural KG completion at the embedding level) mitigate this in some settings but inherit data-quality concerns from the statistical side. Differentiable reasoners such as Neural Logic Machines learn parameterized templates rather than

discrete rules, which sidesteps some misspecification issues but introduces interpretation costs of its own.

Failure modes of hybrid systems. Hybrid architectures introduce failure modes that neither pure symbolic nor pure neural systems face on their own. *Reasoning shortcuts* occur when a neuro-symbolic system computes the right answer for the wrong reason, exploiting spurious associations in the perceptual front-end while the symbolic back-end remains nominally correct; the phenomenon is formalized in [79], which also reports mitigation strategies along with their limitations. A sufficient condition can be stated compactly.

Proposition 7 (A condition for reasoning shortcuts). *Consider a neuro-symbolic system in which inputs x have ground-truth concepts $g^*(x) \in \mathcal{C}$ and labels are produced by fixed symbolic knowledge K via $y = K(g^*(x))$. If there exists a map $\sigma : \mathcal{C} \rightarrow \mathcal{C}$ that is not the identity on the support of g^* and satisfies $K(\sigma(c)) = K(c)$ for every c in that support, then the extractor $\sigma \circ g^*$ attains the same label likelihood as g^* while predicting incorrect concepts on a set of positive probability. Label-based training alone therefore cannot distinguish the two, and every deterministic reasoning shortcut induces such a K -preserving transformation on the support of the ground-truth concepts; see [79] for the general treatment.*

Proof. If σ exists, then $K(\sigma(g^*(x))) = K(g^*(x)) = y$ for every input, so both extractors induce the same conditional label distribution and the same likelihood; conversely, a deterministic shortcut g that reproduces all labels while differing from g^* defines the required transformation by mapping $g^*(x)$ to $g(x)$ wherever g is constant on the level sets of g^* . $\square$

A concrete instance: for the parity task $K(c_1, c_2) = (c_1 + c_2) \bmod 2$ over digits, the shift $\sigma(c) = (c + 2) \bmod 10$ preserves K , so an extractor that systematically confuses each digit with the digit two greater achieves perfect label accuracy while every concept prediction is wrong. Proposition 7 also explains the reported mitigations: concept supervision, multi-task knowledge, and denser label structure all shrink the family of K -preserving transformations toward the identity. *Symbolic–neural interface brittleness*, drift in the schema, vocabulary, or grammar at the boundary of the two components during training or deployment, is a less formally studied but practically common failure pattern, frequently mentioned in deployment retrospectives and closely related to the shortcut analysis above. *Solver-in-the-loop overhead* is a practical concern for search–verification architectures: every rejected

candidate triggers another generation cycle, and the resulting latency and cost can dominate the system, as the LLM-Modulo position itself flags, [3]. None of these failure modes invalidates hybrid designs; all of them argue for dedicated evaluation protocols that go beyond end-to-end task accuracy.

Open challenges. Each of the limitations above translates into a research direction (knowledge acquisition, error bounds across learner and solver, shortcut diagnostics, and contestability metrics), taken up in concrete form by the future-work agenda of Section 9.2.

9 Discussion, Recommendations, and Future Work

The evidence assembled here supports a measured conclusion: knowledge-based methods continue to play a productive role in the era of large language models, particularly for tasks defined by combinatorial structure, declarative correctness criteria, strong priors, data efficiency, and the need for auditable reasoning. Equally, the evidence does not support a paradigmatic narrative of symbolic methods supplanting statistical learning or vice versa. The most productive framing is the task-and-architecture one adopted in Section 4: which problem characteristics call for symbolic representations, which call for statistical learning, and which call for hybrid systems. The formal analysis of Section 7 sharpens that framing into quantitative form: verifier soundness, not generator quality, controls the correctness of search–verification systems (Corollary 1); the constraint family determines the tractability of constraint-guided decoding (Proposition 4); and Inequality (4) states exactly when the cost of a solver in the loop is worth paying.

9.1 Recommendations for practice

Match architecture to task characteristics.

Practitioners selecting an architecture for a new problem can use Table 1 (Appendix) as a starting point: identify which task characteristics are present and consult the corresponding symbolic traditions. The integration patterns of Section 6 then suggest how to couple those traditions with a neural component when a hybrid is appropriate: logic regularization or PLP for end-to-end training with constraints, constraint-guided decoding when the output space has declarative structure, search–verification loops when correctness can be checked, and GraphRAG when grounding in evolving or organization-specific facts is the priority. Defaulting to an end-to-end neural model is appropriate where none of the nine

characteristics apply; conversely, when several apply, a hybrid configuration will usually outperform pure neural baselines.

Adopt the standard integration patterns where appropriate. The five patterns of Section 6 (logic regularization, constraint-guided decoding, search–verification loops (LLM-Modulo), graph retrieval (GraphRAG), and probabilistic logic programming) each have growing empirical evidence and operational tooling. Modular designs in which neural components propose and symbolic components verify, retrieve, or constrain are the dominant practical pattern.

Treat knowledge engineering as a first-class concern. The cost of constructing and maintaining symbolic knowledge remains a bottleneck (Section 8). Combining inductive logic programming, knowledge-graph construction, and LLM-assisted ontology engineering can lower this cost; SHACL-style validation can quickly surface inconsistencies in machine-extracted graphs, [39], [67].

Build for auditability. As AI regulation matures, concept-based and logic-explained networks, argumentation frameworks, and rule extraction together offer a toolkit for making decisions inspectable, [34], [35], [54], [55], [68]. These tools are likely to transition from research prototypes to operational requirements.

9.2 Future work

Six directions stand out as priorities that the present evidence base does not yet adequately cover.

Broader benchmark suites. Existing reasoning evaluations are narrow. Future suites should couple perception with structured reasoning, include real-world planning, scheduling, and verification problems, and report metrics beyond accuracy: constraint violations, optimality gaps, proof lengths, faithfulness of explanations, hallucination rates, and reasoning-shortcut probes, [79], [80].

Theory and tooling for hybrid systems. Search–verification loops and GraphRAG-style architectures need theoretical foundations that account for joint error of the learner and the solver, principled cost–accuracy trade-offs, and standardized tooling that lowers the engineering cost of solver-in-the-loop deployments. Section 7 contributes first-order models of exactly this kind (Propositions 1, 2, 4, and 5); extending them

to correlated candidate draws, adaptive iteration schedules, and PAC-style joint generalization bounds over learner and solver is the natural next step.

Automatic acquisition of symbolic knowledge. Coupling inductive logic programming with LLM-assisted extraction and human-in-the-loop validation is a promising recipe for lowering the knowledge-engineering bottleneck. SHACL-based feedback on machine-extracted KGs is a concrete near-term opportunity.

Causal reasoning at the neuro-symbolic boundary. As Section 5 acknowledges, the evidence base for neuro-symbolic causal discovery is still thin. Bridging structural causal models, knowledge graph embeddings, and LLM-assisted variable identification is an open research program.

Energy and sustainability. A common claim is that symbolic algorithms are computationally light by comparison with large neural models. The reality is more nuanced: linear-time CDCL on easy instances and tractable description-logic reasoning are indeed cheap, but full OWL reasoning, large MaxSAT solves, model checking, and ASP grounding on real-world programs can be expensive in their own right. Where the trade-off favors symbolic methods, hybrid designs that route easy sub-problems to a lightweight symbolic component and reserve neural inference for the genuinely perceptual or generative parts can reduce energy use and serving costs. Quantifying these trade-offs in standard benchmarks would help the community make informed engineering choices, [81].

Interdisciplinary collaboration. Progress requires cooperation across machine learning, knowledge representation, the Semantic Web community, cognitive science, and the application domains where these systems are deployed.

10 Conclusion

Across twelve problem classes, from satisfiability and constraint satisfaction at one end to formal argumentation, Semantic Web reasoning, and concept-based interpretability at the other, the evidence assembled here shows that knowledge-based and neuro-symbolic systems retain measurable advantages on tasks characterized by declarative correctness criteria, combinatorial structure, strong priors, data efficiency, or auditable reasoning. This evidence is not a verdict against statistical learning, which is the right tool for a wide range of problems, including those it has come to

dominate, and which has produced an unprecedented level of competence in language understanding and approximate reasoning. The proper synthesis is the task-and-architecture analysis articulated in Sections 4 and 6 and formalized in Section 7: match each problem to the methods best suited to its characteristics, and combine them through the five integration patterns surveyed here, namely logic regularization, constraint-guided decoding, search-verification loops (LLM-Modulo), graph retrieval (GraphRAG), and probabilistic logic programming. The formal analysis makes the architectural advice quantitative: error propagation bounds and the verification inequality of Section 7 indicate to a practitioner when a verifier in the loop justifies its cost, and the complexity results delimit which constraint families admit tractable enforcement at decoding time. The unifying view across these patterns is that symbolic structure serves as an inductive bias, supplying what statistical learning cannot readily infer from data alone: constraints, schemas, ontologies, and proof obligations that improve data efficiency, robustness, and auditability. Looking ahead, the most productive direction is not to declare a winner among paradigms but to invest in benchmarks, theory, and tooling that make hybrid systems easier to build, evaluate, and trust; the future-work agenda of Section 9.2 sketches a concrete program along those lines.

References:

- [1] R. Bommasani *et al.*, “On the Opportunities and Risks of Foundation Models,” *arXiv preprint arXiv:2108.07258*, 2021. [Online]. Available: <https://doi.org/10.48550/arXiv.2108.07258>
- [2] J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler, E. H. Chi, T. Hashimoto, O. Vinyals, P. Liang, J. Dean, and W. Fedus, “Emergent Abilities of Large Language Models,” *Transactions on Machine Learning Research*, 2022, [Access Date: August 12, 2026]. [Online]. Available: <https://openreview.net/forum?id=yzkSU5zdwD>
- [3] S. Kambhampati, K. Valmeekam, L. Guan, M. Verma, K. Stechly, S. Bhambri, L. Saldyt, and A. Murthy, “Position: LLMs Can’t Plan, But Can Help Planning in LLM-Modulo Frameworks,” in *Proceedings of the 41st International Conference on Machine Learning (ICML)*, ser. Proceedings of Machine Learning Research, vol. 235, Vienna, Austria, 2024, pp. 22 895–22 907, [Access Date: August 12, 2026]. [Online]. Available: <https://proceedings.mlr.press/v235/kambhampati24a.html>
- [4] S. Kambhampati, “Can Large Language Models Reason and Plan?” *Annals of the New York Academy of Sciences*, vol. 1534, no. 1, pp. 15–18, 2024. [Online]. Available: <https://doi.org/10.1111/nyas.15125>
- [5] K. Hammond and D. Leake, “Large Language Models Need Symbolic AI,” in *Proceedings of the 17th International Workshop on Neural-Symbolic Learning and Reasoning (NeSy)*, CEUR Vol-3432, Siena, Italy, 2023, pp. 204–209, [Access Date: August 12, 2026]. [Online]. Available: <https://ceur-ws.org/Vol-3432/paper17.pdf>
- [6] K. Stechly, K. Valmeekam, and S. Kambhampati, “Chain of Thoughtlessness? An Analysis of CoT in Planning,” in *Advances in Neural Information Processing Systems 37 (NeurIPS 2024)*, Vancouver, Canada, 2024, pp. 29 106–29 141. [Online]. Available: <https://doi.org/10.48550/arXiv.2405.04776>
- [7] K. Valmeekam, K. Stechly, A. Gundawar, and S. Kambhampati, “A Systematic Evaluation of the Planning and Scheduling Abilities of the Reasoning Model o1,” *Transactions on Machine Learning Research*, 2025, [Access Date: August 12, 2026]. [Online]. Available: <https://openreview.net/forum?id=FkKBxp0FhR>
- [8] K. Valmeekam, M. Marquez, A. Olmo, S. Sreedharan, and S. Kambhampati, “PlanBench: An Extensible Benchmark for Evaluating Large Language Models on Planning and Reasoning about Change,” in *Advances in Neural Information Processing Systems 36 (NeurIPS 2023)*, Datasets and Benchmarks Track, New Orleans, LA, USA, 2023, [Access Date: August 12, 2026]. [Online]. Available: <https://openreview.net/forum?id=YXogl4uQUO>
- [9] G. Marcus, “The Next Decade in AI: Four Steps Towards Robust Artificial Intelligence,” *arXiv preprint arXiv:2002.06177*, 2020. [Online]. Available: <https://doi.org/10.48550/arXiv.2002.06177>
- [10] A. d’Avila Garcez and L. C. Lamb, “Neurosymbolic AI: The 3rd Wave,” *Artificial Intelligence Review*, vol. 56, no. 11, pp. 12 387–12 406, 2023. [Online]. Available: <https://doi.org/10.1007/s10462-023-10448-w>

- [11] P. Hitzler, A. Eberhart, M. Ebrahimi, M. K. Sarker, and L. Zhou, "Neuro-Symbolic Approaches in Artificial Intelligence," *National Science Review*, vol. 9, no. 6, p. nwac035, 2022. [Online]. Available: <https://doi.org/10.1093/nsr/nwac035>
- [12] G. P. Delvecchio, L. Molfetta, and G. Moro, "Neuro-Symbolic Artificial Intelligence: A Task-Directed Survey in the Black-Box Models Era," in *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence (IJCAI)*, Montreal, Canada, 2025, pp. 10 418–10 426. [Online]. Available: <https://doi.org/10.24963/ijcai.2025/1157>
- [13] T. Miller, "Explanation in Artificial Intelligence: Insights from the Social Sciences," *Artificial Intelligence*, vol. 267, pp. 1–38, 2019. [Online]. Available: <https://doi.org/10.1016/j.artint.2018.07.007>
- [14] M. Davis and H. Putnam, "A Computing Procedure for Quantification Theory," *Journal of the ACM*, vol. 7, no. 3, pp. 201–215, 1960. [Online]. Available: <https://doi.org/10.1145/321033.321034>
- [15] J. Marques-Silva and K. A. Sakallah, "GRASP: A Search Algorithm for Propositional Satisfiability," *IEEE Transactions on Computers*, vol. 48, no. 5, pp. 506–521, 1999. [Online]. Available: <https://doi.org/10.1109/12.769433>
- [16] S. Malik and L. Zhang, "Boolean Satisfiability from Theoretical Hardness to Practical Success," *Communications of the ACM*, vol. 52, no. 8, pp. 76–82, 2009. [Online]. Available: <https://doi.org/10.1145/1536616.1536637>
- [17] J. K. Fichte, D. L. Berre, M. Hecher, and S. Szeider, "The Silent (R)evolution of SAT," *Communications of the ACM*, vol. 66, no. 6, pp. 64–72, 2023. [Online]. Available: <https://doi.org/10.1145/3560469>
- [18] L. de Moura and N. Bjørner, "Satisfiability Modulo Theories: Introduction and Applications," *Communications of the ACM*, vol. 54, no. 9, pp. 69–77, 2011. [Online]. Available: <https://doi.org/10.1145/1995376.1995394>
- [19] L. de Moura and N. Bjørner, "Z3: An Efficient SMT Solver," in *Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2008)*, LNCS 4963. Budapest, Hungary: Springer, 2008, pp. 337–340. [Online]. Available: https://doi.org/10.1007/978-3-540-78800-3_24
- [20] R. Soto, B. Crawford, C. Galleguillos, F. Paredes, and E. Norero, "A Hybrid alldifferent–Tabu Search Algorithm for Solving Sudoku Puzzles," *Computational Intelligence and Neuroscience*, vol. 2015, no. 286354, pp. 1–10, 2015. [Online]. Available: <https://doi.org/10.1155/2015/286354>
- [21] M. Bofill, J. Coll, M. Garcia, J. Giráldez-Cru, G. Pesant, J. Suy, and M. Villaret, "Constraint Solving Approaches to the Business-to-Business Meeting Scheduling Problem," *Journal of Artificial Intelligence Research*, vol. 74, pp. 263–301, 2022. [Online]. Available: <https://doi.org/10.1613/jair.1.12670>
- [22] A. L. Blum and M. L. Furst, "Fast Planning through Planning Graph Analysis," *Artificial Intelligence*, vol. 90, no. 1–2, pp. 281–300, 1997. [Online]. Available: [https://doi.org/10.1016/S0004-3702\(96\)00047-1](https://doi.org/10.1016/S0004-3702(96)00047-1)
- [23] D. Speck, J. Seipp, and Á. Torralba, "Symbolic Search for Cost-Optimal Planning with Expressive Model Extensions," *Journal of Artificial Intelligence Research*, vol. 82, pp. 1349–1405, 2025. [Online]. Available: <https://doi.org/10.1613/jair.1.16869>
- [24] S. Muggleton and L. D. Raedt, "Inductive Logic Programming: Theory and Methods," *The Journal of Logic Programming*, vol. 19–20, pp. 629–679, 1994. [Online]. Available: [https://doi.org/10.1016/0743-1066\(94\)90035-3](https://doi.org/10.1016/0743-1066(94)90035-3)
- [25] H. Dong, J. Mao, T. Lin, C. Wang, L. Li, and D. Zhou, "Neural Logic Machines," in *International Conference on Learning Representations (ICLR)*, New Orleans, LA, USA, 2019. [Online]. Available: <https://doi.org/10.48550/arXiv.1904.11694>
- [26] G. Brewka, T. Eiter, and M. Truszczynski, "Answer Set Programming at a Glance," *Communications of the ACM*, vol. 54, no. 12, pp. 92–103, 2011. [Online]. Available: <https://doi.org/10.1145/2043174.2043195>
- [27] T. Eiter, G. Ianni, and T. Krennwallner, "Answer Set Programming: A Primer," in *Reasoning Web. Semantic Technologies for Information Systems (LNCS 5689)*. Berlin, Heidelberg: Springer, 2009, pp. 40–110, iISBN: 978-3-642-03753-5. [Online]. Available: https://doi.org/10.1007/978-3-642-03754-2_2

- [28] J. Fandinno and C. Schulz, “Answering the “Why” in Answer Set Programming – A Survey of Explanation Approaches,” *Theory and Practice of Logic Programming*, vol. 19, no. 2, pp. 114–203, 2019. [Online]. Available: <https://doi.org/10.1017/S1471068418000534>
- [29] A. Skryagin, D. Ochs, D. S. Dhami, and K. Kersting, “Scalable Neural-Probabilistic Answer Set Programming,” *Journal of Artificial Intelligence Research*, vol. 78, pp. 579–617, 2023. [Online]. Available: <https://doi.org/10.1613/jair.1.15027>
- [30] T. Eiter, N. H. Ruiz, and J. Oetsch, “A Modular Neurosymbolic Approach for Visual Graph Question Answering,” in *Proceedings of the 17th International Workshop on Neural-Symbolic Learning and Reasoning (NeSy)*, CEUR Vol-3432, Siena, Italy, 2023, pp. 139–149, [Access Date: August 12, 2026]. [Online]. Available: <https://ceur-ws.org/Vol-3432/paper11.pdf>
- [31] P. M. Dung, “On the Acceptability of Arguments and Its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n -Person Games,” *Artificial Intelligence*, vol. 77, no. 2, pp. 321–357, 1995. [Online]. Available: [https://doi.org/10.1016/0004-3702\(94\)00041-X](https://doi.org/10.1016/0004-3702(94)00041-X)
- [32] K. Atkinson, P. Baroni, M. Giacomini, A. Hunter, H. Prakken, C. Reed, G. R. Simari, M. Thimm, and S. Villata, “Toward Artificial Argumentation,” *AI Magazine*, vol. 38, no. 3, pp. 25–36, 2017. [Online]. Available: <https://doi.org/10.1609/aimag.v38i3.2704>
- [33] M. Ulbricht, “Formal Argumentation in Symbolic AI,” in *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence (IJCAI), Early Career Track*, Jeju, South Korea, 2024, pp. 8577–8582. [Online]. Available: <https://doi.org/10.24963/ijcai.2024/985>
- [34] A. Vassiliades, N. Bassiliades, and T. Patkos, “Argumentation and Explainable Artificial Intelligence: A Survey,” *The Knowledge Engineering Review*, vol. 36, p. e5, 2021. [Online]. Available: <https://doi.org/10.1017/S0269888921000011>
- [35] K. Čyras, A. Rago, E. Albini, P. Baroni, and F. Toni, “Argumentative XAI: A Survey,” in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI), Survey Track*, Montreal, Canada (virtual), 2021, pp. 4392–4399. [Online]. Available: <https://doi.org/10.24963/ijcai.2021/600>
- [36] F. Castagna, N. Kökciyan, I. Sassoon, S. Parsons, and E. Sklar, “Computational Argumentation-Based Chatbots: A Survey,” *Journal of Artificial Intelligence Research*, vol. 80, pp. 1271–1310, 2024. [Online]. Available: <https://doi.org/10.1613/jair.1.15407>
- [37] W3C OWL Working Group, “OWL 2 Web Ontology Language Document Overview (Second Edition),” W3C Recommendation of December 11, 2012, [Access Date: August 12, 2026]. [Online]. Available: <https://www.w3.org/TR/owl2-overview/>
- [38] S. Harris and A. Seaborne, “SPARQL 1.1 Query Language,” W3C Recommendation of March 21, 2013, [Access Date: August 12, 2026]. [Online]. Available: <https://www.w3.org/TR/sparql11-query/>
- [39] H. Knublauch and D. Kontokostas, “Shapes Constraint Language (SHACL),” W3C Recommendation of July 20, 2017, [Access Date: August 12, 2026]. [Online]. Available: <https://www.w3.org/TR/shacl/>
- [40] A. Hogan, E. Blomqvist, M. Cochez, C. d’Amato, G. de Melo, C. Gutierrez, S. Kirrane, J. E. L. Gajo, R. Navigli, S. Neumaier, A.-C. N. Ngomo, A. Polleres, S. M. Rashid, A. Rula, L. Schmelzeisen, J. Sequeda, S. Staab, and A. Zimmermann, “Knowledge Graphs,” *ACM Computing Surveys*, vol. 54, no. 4, pp. 71:1–71:37, 2021. [Online]. Available: <https://doi.org/10.1145/3447772>
- [41] D. Herron, E. Jiménez-Ruiz, and T. Weyde, “On the Potential of Logic and Reasoning in Neurosymbolic Systems Using OWL-Based Knowledge Graphs,” *Neurosymbolic Artificial Intelligence*, vol. 1, 2025. [Online]. Available: <https://doi.org/10.1177/29498732251320043>
- [42] 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. [Online]. Available: <https://doi.org/10.1109/TPAMI.2024.3483273>
- [43] L. N. DeLong, R. F. Mir, and J. D. Fleuriot, “Neurosymbolic AI for Reasoning over Knowledge Graphs: A Survey,” *IEEE Transactions on Neural Networks*

and Learning Systems, vol. 36, no. 5, pp. 7822–7842, 2025. [Online]. Available: <https://doi.org/10.1109/TNNLS.2024.3420218>

- [44] K. Cheng, N. K. Ahmed, R. A. Rossi, T. L. Willke, and Y. Sun, “Neural-Symbolic Methods for Knowledge Graph Reasoning: A Survey,” *ACM Transactions on Knowledge Discovery from Data*, vol. 18, no. 9, pp. 225:1–225:44, 2024. [Online]. Available: <https://doi.org/10.1145/3686806>
- [45] Y. Xu, G. Liu, R. R. Kompella, S. Hu, F. Ilhan, S. F. Tekin, Z. Yahn, and L. Liu, “A Neurosymbolic Agent System for Compositional Visual Reasoning,” *arXiv preprint arXiv:2506.07778*, 2025. [Online]. Available: <https://doi.org/10.48550/arXiv.2506.07778>
- [46] Z. Tan, X. Yang, Q. Wang, A. Nguyen, and K. Huang, “Interpret Your Decision: Logical Reasoning Regularization for Generalization in Visual Classification,” in *Advances in Neural Information Processing Systems 37 (NeurIPS 2024)*, Vancouver, Canada, 2024, pp. 18 166–18 204. [Online]. Available: <https://doi.org/10.48550/arXiv.2410.04492>
- [47] B. Peng, Y. Zhu, Y. Liu, X. Bo, H. Shi, C. Hong, Y. Zhang, and S. Tang, “Graph Retrieval-Augmented Generation: A Survey,” *ACM Transactions on Information Systems*, vol. 44, no. 2, pp. 1–52, 2026. [Online]. Available: <https://doi.org/10.1145/3777378>
- [48] H. Han, Y. Wang, H. Shomer, K. Guo, J. Ding, Y. Lei, M. Halappanavar, R. A. Rossi, S. Mukherjee, X. Tang, Q. He, Z. Hua, B. Long, T. Zhao, N. Shah, A. Javari, Y. Xia, and J. Tang, “Retrieval-Augmented Generation with Graphs (GraphRAG),” *arXiv preprint arXiv:2501.00309*, 2025. [Online]. Available: <https://doi.org/10.48550/arXiv.2501.00309>
- [49] D. Edge, H. Trinh, N. Cheng, J. Bradley, A. Chao, A. Mody, S. Truitt, D. Metropolitansky, R. O. Ness, and J. Larson, “From Local to Global: A Graph RAG Approach to Query-Focused Summarization,” *arXiv preprint arXiv:2404.16130*, 2024. [Online]. Available: <https://doi.org/10.48550/arXiv.2404.16130>
- [50] M. Leon, N. M. Sanchez, Z. G. Valdivia, and R. B. Perez, “Concept Maps Combined with Case-Based Reasoning in Order to Elaborate Intelligent Teaching/Learning Systems,” in *Seventh International Conference on Intelligent Systems Design and Applications (ISDA 2007)*. Rio de Janeiro, Brazil: IEEE, Oct. 2007, pp. 205–210. [Online]. Available: <https://doi.org/10.1109/ISDA.2007.33>
- [51] L. Li, W. Wang, and Y. Yang, “LogicSeg: Parsing Visual Semantics with Neural Logic Learning and Reasoning,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, Paris, France, 2023, pp. 4099–4110. [Online]. Available: <https://doi.org/10.1109/ICCV51070.2023.00381>
- [52] L. Li, J. Wei, W. Wang, and Y. Yang, “Neural-Logic Human-Object Interaction Detection,” in *Advances in Neural Information Processing Systems 36 (NeurIPS 2023)*, New Orleans, LA, USA, 2023, pp. 21 158–21 171. [Online]. Available: <https://doi.org/10.48550/arXiv.2311.09817>
- [53] Z. Yang, A. Ishay, and J. Lee, “NeurASP: Embracing Neural Networks into Answer Set Programming,” in *Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI)*, Yokohama, Japan (held virtually), 2020, pp. 1755–1762. [Online]. Available: <https://doi.org/10.24963/ijcai.2020/243>
- [54] P. Barbiero, G. Ciravegna, F. Giannini, M. E. Zarlenga, L. C. Magister, A. Tonda, P. Lio, F. Precioso, M. Jamnik, and G. Marra, “Interpretable Neural-Symbolic Concept Reasoning,” in *Proceedings of the 40th International Conference on Machine Learning (ICML)*, ser. Proceedings of Machine Learning Research, vol. 202, Honolulu, HI, USA, 2023, pp. 1801–1825, [Access Date: August 12, 2026]. [Online]. Available: <https://proceedings.mlr.press/v202/barbiero23a.html>
- [55] G. Ciravegna, P. Barbiero, F. Giannini, M. Gori, P. Lió, M. Maggini, and S. Melacci, “Logic Explained Networks,” *Artificial Intelligence*, vol. 314, p. 103822, 2023. [Online]. Available: <https://doi.org/10.1016/j.artint.2022.103822>
- [56] P. W. Koh, T. Nguyen, Y. S. Tang, S. Mussmann, E. Pierson, B. Kim, and P. Liang, “Concept Bottleneck Models,” in *Proceedings of the 37th International Conference on Machine Learning (ICML)*, ser. Proceedings of Machine Learning Research, vol. 119, Virtual Event, 2020, pp. 5338–5348, [Access Date: August 12, 2026]. [Online]. Available: <https://proceedings.mlr.press/v119/koh20a.html>

- [57] K. Schreiberhuber, M. Sabou, F. J. Ekaputra, P. Knees, P. R. Aryan, A. Einfalt, and R. Mosshammer, "Causality Prediction with Neural-Symbolic Systems: A Case Study in Smart Grids," in *Proceedings of the 17th International Workshop on Neural-Symbolic Learning and Reasoning (NeSy)*, CEUR Vol-3432, Siena, Italy, 2023, pp. 336–347, [Access Date: August 12, 2026]. [Online]. Available: <https://ceur-ws.org/Vol-3432/paper29.pdf>
- [58] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," in *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, vol. 33, Virtual Event, 2020, pp. 9459–9474, [Access Date: August 12, 2026]. [Online]. Available: <https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html>
- [59] J. Sun, C. Xu, L. Tang, S. Wang, C. Lin, Y. Gong, L. M. Ni, H. Shum, and J. Guo, "Think-on-Graph: Deep and Responsible Reasoning of Large Language Model on Knowledge Graph," in *International Conference on Learning Representations (ICLR)*, Vienna, Austria, 2024, [Access Date: August 12, 2026]. [Online]. Available: <https://openreview.net/forum?id=nnVO1PvbTv>
- [60] F. Chollet, "On the Measure of Intelligence," *arXiv preprint arXiv:1911.01547*, 2019. [Online]. Available: <https://doi.org/10.48550/arXiv.1911.01547>
- [61] F. Chollet, M. Knoop, G. Kamradt, and B. Landers, "ARC Prize 2024: Technical Report," *arXiv preprint arXiv:2412.04604*, 2024. [Online]. Available: <https://doi.org/10.48550/arXiv.2412.04604>
- [62] T. H. Trinh, Y. Wu, Q. V. Le, H. He, and T. Luong, "Solving Olympiad Geometry without Human Demonstrations," *Nature*, vol. 625, no. 7995, pp. 476–482, 2024. [Online]. Available: <https://doi.org/10.1038/s41586-023-06747-5>
- [63] S. Sinha, A. Prabhu, P. Kumaraguru, S. Bhat, and M. Bethge, "Wu's Method Can Boost Symbolic AI to Rival Silver Medalists and AlphaGeometry to Outperform Gold Medalists at IMO Geometry," *arXiv preprint arXiv:2404.06405*, 2024. [Online]. Available: <https://doi.org/10.48550/arXiv.2404.06405>
- [64] Y. Chervonyi, T. H. Trinh, M. Olšák, X. Yang, H. Nguyen, M. Menegali, J. Jung, J. Kim, V. Verma, Q. V. Le, and T. Luong, "Gold-Medalist Performance in Solving Olympiad Geometry with AlphaGeometry2," *Journal of Machine Learning Research*, vol. 26, no. 241, pp. 1–39, 2025, [Access Date: August 12, 2026]. [Online]. Available: <https://jmlr.org/papers/v26/25-1654.html>
- [65] P. Cao, T. Men, W. Liu, J. Zhang, X. Li, X. Lin, D. Sui, Y. Cao, K. Liu, and J. Zhao, "Large Language Models for Planning: A Comprehensive and Systematic Survey," *arXiv preprint arXiv:2505.19683*, 2025. [Online]. Available: <https://doi.org/10.48550/arXiv.2505.19683>
- [66] M. Leon, "Lifecycle-Based Governance to Build Reliable Ethical AI Systems," *Systems Research and Behavioral Science*, vol. 43, no. 3, pp. 1116–1132, 2026. [Online]. Available: <https://doi.org/10.1002/sres.70014>
- [67] P. Pareti and G. Konstantinidis, "A Review of SHACL: From Data Validation to Schema Reasoning for RDF Graphs," in *Reasoning Web. Declarative Artificial Intelligence (LNCS 13100)*. Cham: Springer, 2022, pp. 115–144, ISBN: 978-3-030-95480-2. [Online]. Available: https://doi.org/10.1007/978-3-030-95481-9_6
- [68] R. Andrews, J. Diederich, and A. B. Tickle, "Survey and Critique of Techniques for Extracting Rules from Trained Artificial Neural Networks," *Knowledge-Based Systems*, vol. 8, no. 6, pp. 373–389, 1995. [Online]. Available: [https://doi.org/10.1016/0950-7051\(96\)81920-4](https://doi.org/10.1016/0950-7051(96)81920-4)
- [69] G. Bologna and Y. Hayashi, "Characterization of Symbolic Rules Embedded in Deep DIMLP Networks: A Challenge to Transparency of Deep Learning," *Journal of Artificial Intelligence and Soft Computing Research*, vol. 7, no. 4, pp. 265–286, 2017. [Online]. Available: <https://doi.org/10.1515/jaiscr-2017-0019>
- [70] S. E. Mekkaoui, L. Benabbou, and A. Berrado, "Rule-Extraction Methods From Feedforward Neural Networks: A Systematic Literature Review," *arXiv preprint arXiv:2312.12878*, 2023. [Online]. Available: <https://doi.org/10.48550/arXiv.2312.12878>

- [71] C. Demir and A. N. Ngomo, "Neuro-Symbolic Class Expression Learning," in *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence (IJCAI)*, Macao, SAR, China, 2023, pp. 3624–3632. [Online]. Available: <https://doi.org/10.24963/ijcai.2023/403>
- [72] J. Xu, Z. Zhang, T. Friedman, Y. Liang, and G. Van den Broeck, "A Semantic Loss Function for Deep Learning with Symbolic Knowledge," in *Proceedings of the 35th International Conference on Machine Learning (ICML)*, ser. Proceedings of Machine Learning Research, vol. 80, Stockholm, Sweden, 2018, pp. 5502–5511, [Access Date: August 12, 2026]. [Online]. Available: <https://proceedings.mlr.press/v80/xu18h.html>
- [73] L. D. Raedt, A. Kimmig, and H. Toivonen, "ProbLog: A Probabilistic Prolog and its Application in Link Discovery," in *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, Hyderabad, India, 2007, pp. 2468–2473, [Access Date: August 12, 2026]. [Online]. Available: <https://www.ijcai.org/Proceedings/07/Papers/396.pdf>
- [74] R. Manhaeve, S. Dumančić, A. Kimmig, T. Demeester, and L. D. Raedt, "DeepProbLog: Neural Probabilistic Logic Programming," in *Advances in Neural Information Processing Systems 31 (NeurIPS 2018)*, Montreal, Canada, 2018, pp. 3753–3763, [Access Date: August 12, 2026]. [Online]. Available: <https://proceedings.neurips.cc/paper/2018/hash/dc5d637ed5e62c36ecb73b654b05ba2a-Abstract.html>
- [75] S. Ghadimi and G. Lan, "Stochastic First- and Zeroth-Order Methods for Nonconvex Stochastic Programming," *SIAM Journal on Optimization*, vol. 23, no. 4, pp. 2341–2368, 2013. [Online]. Available: <https://doi.org/10.1137/120880811>
- [76] D. P. Bertsekas, *Nonlinear Programming*, 2nd ed. Belmont, MA: Athena Scientific, 1999, ISBN: 978-1-886529-00-7. [Access Date: August 12, 2026]. [Online]. Available: <http://www.athenasc.com/nonlinbook.html>
- [77] S. A. Cook, "The Complexity of Theorem-Proving Procedures," in *Proceedings of the Third Annual ACM Symposium on Theory of Computing (STOC)*, Shaker Heights, OH, USA, 1971, pp. 151–158. [Online]. Available: <https://doi.org/10.1145/800157.805047>
- [78] M. Leon, "Generative AI as a General-Purpose Technology: Foundations, Applications, and Labor Market Implications Through 2030," *Big Data and Cognitive Computing*, vol. 10, no. 3, p. 69, 2026. [Online]. Available: <https://doi.org/10.3390/bdcc10030069>
- [79] E. Marconato, S. Teso, and A. Passerini, "Neuro-Symbolic Reasoning Shortcuts: Mitigation Strategies and Their Limitations," in *Proceedings of the 17th International Workshop on Neural-Symbolic Learning and Reasoning (NeSy)*, CEUR Vol-3432, Siena, Italy, 2023, pp. 162–166, [Access Date: August 12, 2026]. [Online]. Available: <https://ceur-ws.org/Vol-3432/paper13.pdf>
- [80] M. Leon, "Investing in AI Interpretability, Control, and Robustness," *Algorithms*, vol. 19, no. 2, p. 136, 2026. [Online]. Available: <https://doi.org/10.3390/a19020136>
- [81] M. Leon, "Generative Artificial Intelligence and Prompt Engineering: A Comprehensive Guide to Models, Methods, and Best Practices," *Advances in Science, Technology and Engineering Systems Journal*, vol. 10, no. 2, pp. 1–11, 2025. [Online]. Available: <https://doi.org/10.25046/aj100201>

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)

The author contributed in the present research, at all stages from the formulation of the problem to the final findings and solution.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself

No funding was received for conducting this study.

Conflict of Interest

The author has no conflict of interest to declare that is relevant to the content of this article.

Creative Commons Attribution License 4.0 (Attribution 4.0 International, CC BY 4.0)

This article is published under the terms of the Creative Commons Attribution License 4.0

https://creativecommons.org/licenses/by/4.0/deed.en_US

A Glossary of Acronyms

ABA	Assumption-Based Argumentation
ARC	Abstraction and Reasoning Corpus
ASP	Answer Set Programming
CDCL	Conflict-Driven Clause Learning
CP	Constraint Programming
CSP	Constraint Satisfaction Problem
DCR	Deep Concept Reasoner
DFA	Deterministic Finite Automaton
DL	Description Logic
ERM	Empirical Risk Minimization
HOI	Human-Object Interaction
ILP	Inductive Logic Programming
KG	Knowledge Graph
LLM	Large Language Model
mAP	Mean Average Precision
mIoU	Mean Intersection over Union
MILP	Mixed-Integer Linear Programming
OCR	Optical Character Recognition
OWL	Web Ontology Language
PDDL	Planning Domain Definition Language
PLP	Probabilistic Logic Programming
RAG	Retrieval-Augmented Generation
RDF	Resource Description Framework
SAT	Boolean Satisfiability
SHACL	Shapes Constraint Language
SMT	Satisfiability Modulo Theories
SPARQL	SPARQL Protocol and RDF Query Language
VQA	Visual Question Answering
XAI	Explainable Artificial Intelligence

B Full-Width Tables

This appendix collects the two tables that span the full page width: the mapping from task characteristics to symbolic traditions (Table 1) and the summary of empirical comparisons (Table 2).

Table 1: Mapping from task characteristics (Section 4) to the symbolic traditions of Section 3 that most directly support them. Each row is one of the nine characteristics introduced in Section 4; each is also revisited in the italicized closing line of the relevant Section 5 subsections and tabulated quantitatively in Table 2. Cells with a ✓ mark a primary fit; multiple ticks per row reflect that several traditions can serve the same characteristic. The mapping is intended as a lookup rather than a partition: traditions may support a characteristic to varying degrees, and a given task may exhibit several characteristics simultaneously.

	<i>SAT/SMT</i>	<i>CP / Comb. opt.</i>	<i>Planning</i>	<i>ILP / synthesis</i>	<i>ASP</i>	<i>Argumentation</i>	<i>Sem. Web / KG</i>
Declarative correctness criteria	✓	✓	✓	✓	✓	✓	✓
Combinatorial structure / search	✓	✓	✓		✓	✓	✓
Strong priors / domain knowledge		✓	✓	✓	✓	✓	✓
Data efficiency				✓	✓		✓
Compositional / systematic generalization				✓	✓		
Interpretability / auditability			✓	✓	✓	✓	✓
Defeasible / contested reasoning					✓	✓	
Causal inference / knowledge discovery				✓	✓		✓
Grounding in evolving / org.-specific facts					✓		✓

Source: created by the authors.

Table 2: Representative empirical comparisons in which symbolic or neuro-symbolic systems retain advantages over end-to-end neural or LLM baselines. The table reports system, baseline, dataset/setting, evaluation metric, and observed result; subsections in this section elaborate on context and caveats. Where the cited source reports a specific numerical result, that number appears in the final column.

Domain	Symbolic / hybrid system	Neural / LLM baseline	Dataset or setting	Metric	Observed result (source)
Satisfiability	CDCL solvers (MiniSAT, Glucose, Kissat)	NeuroSAT, neural local search	SAT competition / industrial instances with 10^5 – 10^7 variables	Solved instances within time limit	CDCL solvers routinely scale to industrial instances with millions of variables; neural SAT solvers remain confined to small instances, [16], [17]
Constraint satisfaction	alldifferent + Tabu search	CNN baselines on Sudoku	9×9 Sudoku puzzles	Accuracy, runtime	100% solve rate with faster runtimes than approximate neural heuristics, [20]
Abstract reasoning & program induction	Neural Logic Machines (NLM); SLASH; neuro-symbolic program synthesis	GPT-4, vanilla Transformers	Algorithmic tasks (sorting, shortest paths); ARC; visual arithmetic	Generalization accuracy	NLM generalizes perfectly to inputs an order of magnitude larger than those seen in training; SLASH improves over neural baselines on visual arithmetic, [25], [29]
Formal mathematics	Coq, Lean, Isabelle; AlphaGeometry (LM + DD+AR)	End-to-end LLM proof generators; Wu's method (fully symbolic)	IMO-AG-30 (30 olympiad geometry problems)	Solved problems	AlphaGeometry: 25/30; Wu's method: 10/30, [13], [62]
Planning & scheduling	Symbolic optimal planners; MaxSAT scheduler; LLM-Modulo	GPT-4, o1, RL planners	PlanBench; business-to-business meeting scheduling	Plan validity, makespan, runtime	Classical planners produce valid optimal plans where LLMs frequently fail; MaxSAT improves on CP and MILP by sometimes an order of magnitude on meeting scheduling, [3], [7], [8], [21], [23]
Compositional generalization	ILP; stack-augmented neuro-symbolic systems	Vanilla Transformers; LLMs without external symbolic support	SCAN, COGS, algorithmic benchmarks	Out-of-distribution accuracy	Symbolic and ILP-style systems generalize from few examples where LLMs and Transformers remain sensitive to prompt and split, [6], [24], [25]
Visual reasoning	LogicSeg; Neural-Logic HOI; NSGRAPH (OGR + OCR + ASP); VLAgent	Strong CNN / Transformer baselines; end-to-end VLMs	ADE20K, Cityscapes (segm.); HICO-DET (HOI); CLEGR (VGQA); compositional VQA	mIoU, mAP, accuracy	LogicSeg improves mIoU on segmentation; Logic-constrained HOI improves mAP and supports zero-shot; NSGRAPH reaches 73% on CLEGR; VLAgent reports accuracy gains over VLMs with auditable steps, [30], [45], [51], [52]
Answer set programming	NeurASP; SLASH; ASP-backed VQA	DeepProbLog; end-to-end neural classifiers	MNIST-Addition, Sudoku, visual arithmetic	Accuracy, per-iteration training time	NeurASP reaches accuracy comparable to DeepProbLog with notably reduced per-iteration training time on MNIST-Addition and Sudoku; SLASH improves stability and accuracy on visual arithmetic, [29], [53]
Causal inference	KG embeddings + rule-based reasoning	Pure KG embedding baselines	Smart-grid causality prediction	Hits@5	Hybrid predicts causal links absent from rule base, raising Hits@5 (single-case study), [57]
Argumentation & XAI	Abstract / structured argumentation frameworks (abstract AFs, ASPIC ⁺ , ABA)	Black-box neural classifiers; opaque generation	Decision-support, legal, and conversational tasks	Explanation quality, dialogue success	Argumentation yields dialectical explanations and consistent reasoning under conflicting evidence, [34], [35], [36]
Knowledge-graph QA	GraphRAG (graph-based retrieval over a KG)	Vanilla RAG, standalone LLMs	Multi-hop QA, query-focused summarization	Faithfulness, recall, hallucination rate	GraphRAG improves multi-hop reasoning and reduces hallucinations over standalone LLMs and flat text-based RAG, [47], [48], [49], [59]
Interpretability (concept / rule-based)	DCR; logic-explained networks; concept bottleneck models	Black-box classifiers; post-hoc saliency methods	Tabular and image-classification benchmarks with ground-truth concept/rule structure	Accuracy + explanation fidelity	Concept-based and logic-explained networks match or exceed comparable interpretable baselines on accuracy while recovering ground-truth rules and supporting human intervention on concepts, [54], [55], [56]

Source: created by the authors.