

Neurosymbolic Inference, Reasoning and Proving in Propositional Logic: A Survey

Nenad Krdzavac¹ and Felix Engel¹

Abstract

Neurosymbolic (NeSy) AI combines neural learning with symbolic reasoning and proving, but reported capabilities do not by themselves provide assurance of correctness. This survey examines NeSy approaches to reasoning and proving based on propositional logic (PL), distinguishing capability from assurance. The analysis covers 26 survey and review papers and 181 publications from the 1940–2026 search period, with 180 quantitatively evaluated. An X–Y–Z taxonomy is introduced and implemented using the Simple Knowledge Organization System (SKOS). Qualitative annotations support a reproducible Python pipeline computing concept frequencies, capability and assurance scores, joint coverage, and annual statistics. Existing survey classifications do not consistently distinguish PL capabilities from supporting correctness evidence. SAT solving and logic expressivity are assigned to 117 and 112 evaluated publications, respectively, compared with 2 for proof validation and 7 for soundness assessment. These findings indicate an imbalance between reported capabilities and documented assurance, without implying that unannotated frameworks are unsound. They motivate greater attention to proof generation, independent validation, and explicit guarantees. The released taxonomy, annotations, and implementation support reproducible comparison and future extensions.

Keywords

Neurosymbolic AI, Propositional Logic, Inference, Reasoning, Proving

Abbreviations

Abbreviation	Explanation
General AI and neurosymbolic terminology	
NeSy	Neurosymbolic
NeSy AI	Neurosymbolic Artificial Intelligence
Logic, forms and satisfiability	
ANF	Algebraic Normal Form
CNF	Conjunctive Normal Form
DNF	Disjunctive Normal Form
FOL	First-Order Logic
PL	Propositional Logic
PMS	Partial Maximum Satisfiability
SAT	Boolean Satisfiability
UNSAT	Boolean Unsatisfiability
SAT solving and testing algorithms	
ATPG	Automatic Test Pattern Generation
CDCL	Conflict-Driven Clause Learning
DPLL	Davis–Putnam–Logemann–Loveland Algorithm
ICA	Imperialist Competitive Algorithm
MILP	Mixed-Integer Linear Programming
ACO	Ant Colony Optimization
CSP	Constraint Satisfaction Problem
Neural network architectures and models	
ANN	Artificial Neural Network
BM	Boltzmann Machine
BPNN	Backpropagation Neural Network
ConvNN	Convolutional Neural Network
ConnectNN	Connectionist Neural Network
DHNN	Discrete Hopfield Neural Network
DNN	Deep Neural Network
FNN	Feedforward Neural Network
GAT	Graph Attention Network
GCNN	Graph Convolutional Neural Network
GCG-NN	Gated Graph Convolutional Neural Network
DHNN-3SAT	Discrete Hopfield Neural Network for 3-Satisfiability
GNN	Graph Neural Network
GRU-NN	Gated Recurrent Unit Neural Network
HCNN	Hypergraph Convolutional Neural Network
HNN	Hopfield Neural Network
HT ANN	Harmony Theoretic Artificial Neural Network
HyperGCN	Hypergraph Convolutional Network
LBM	Logical Boltzmann Machine
KBANN	Knowledge-Based Artificial Neural Network
LSTM	Long Short-Term Memory
LINN	Logic Integrated Neural Network
LNN	Logical Neural Network
LPPH	Lagrange Programming Neural Network
LSTM RNN	Long Short-Term Memory Recurrent Neural Network
MPNN	Message-Passing Neural Network
MHNN	Mutation Hopfield Neural Network
NLN	Neural Logic Network
RBFNN	Radial Basis Function Neural Network
RBM	Restricted Boltzmann Machine
RNN	Recurrent Neural Network
SAANN	Simulated Annealing Artificial Neural Network
SGAT	SAT-Based Graph Attention Network
SNN	Symmetric Neural Network
TenSAT	Temperature-Scaled Neural Network SAT Solver
Tree-LSTM NN	Tree-Structured Long Short-Term Memory Neural Network
Tree-NN	Tree Neural Network
UANN	Unsupervised Artificial Neural Network
LagONN	Lagrange oscillatory neural network
SOM-NN	Self-Organizing Map Neural Network

¹TIB–Leibniz Information Centre for Science and Technology, Hanover, Germany

Corresponding author:
Nenad Krdzavac, TIB–Leibniz Information Centre for Science and Technology, Hanover, Germany.
Email: Nenad.Krdzavac@tib.eu

Learning, meta-heuristics, and hybrid optimization methods	
ESM	Exhaustive Search Method
HNNICA	Hybrid Hopfield Neural Network-Imperialist Competitive Algorithm
RL	Reinforcement Learning
Knowledge organization and data serialization	
SKOS	Simple Knowledge Organization System
TTL	Turtle Syntax
DRAT	Deletion Resolution Asymmetric Tautology
DRUP	Deletion Reverse Unit Propagation

1 Introduction

Neurosymbolic (NeSy) AI combines neural learning with symbolic reasoning and proving, but reported capabilities do not by themselves provide assurance. Existing surveys examine NeSy AI from several perspectives. For example, the survey in Yu et al. (2023) organizes NeSy learning research around challenges, methods, applications, and future directions, while the survey in Marra et al. (2024) compares NeSy and statistical relational AI along dimensions that include logical inference, syntax, semantics, representations, and learning. These perspectives provide a foundation for understanding how neural and symbolic methods interact. However, as our analysis in Section 4 shows, the reviewed classifications do not consistently distinguish the reasoning and proving capabilities of NeSy approaches based on PL from the mechanisms that provide assurance for their results. This survey addresses this gap by examining PL-based NeSy reasoning and proving while explicitly distinguishing capability from assurance.

This distinction matters because representing logical formulas, predicting their satisfiability, constructing satisfying assignments, and producing checkable proofs are different achievements. A framework may perform well on a reasoning benchmark without providing evidence that each reported conclusion is correct. Similarly, a human-interpretable explanation can help users understand a result without establishing its logical validity. The gap addressed here is therefore the lack of a consistent, PL-focused account of both what surveyed frameworks can do and what evidence supports their results. Explainability remains valuable, but it should be examined alongside soundness, proof validation, and certification. Our analysis does not establish that attention to explainability has caused limited coverage of these other properties.

Reasoning and proving capabilities, together with assurance mechanisms, matter when incorrect conclusions or decisions can have serious consequences. Application areas include autonomous transport, medical devices, and industrial control, where failures may cause injury or damage. In such settings, strong empirical performance alone does not establish that a system satisfies a specified safety requirement. For example, Hashemi et al. (2023) present a neurosymbolic approach to verifying neural network controllers against signal temporal logic specifications. When both the controller and the plant model are represented by ReLU networks, their approach reduces verification to an ANN reachability problem. This illustrates how formal verification can complement learned control, although the specifications extend beyond PL. For problems represented in PL, a satisfying assignment can demonstrate that encoded constraints are jointly satisfiable, while a checked UNSAT refutation can establish that a specified

unsafe condition is impossible within the encoded model. For example, when a formula encodes the permitted states of an industrial controller together with an unsafe combination of actuator commands, a checked refutation establishes that no assignment satisfies both within that encoding. If the encoding covers only a bounded execution, the conclusion is restricted to that bound. Proof generation provides evidence for a conclusion, and proof validation checks that evidence. These distinctions help readers assess both what a NeSy framework can do and what supports confidence in its results. This motivation is consistent with the Guaranteed Safe AI perspective of Dalrymple et al. (2024), which combines a world model, a safety specification, and a verifier to support formal safety guarantees. Such guarantees remain conditional on the adequacy of the model, specification, and verification assumptions. Our taxonomy therefore distinguishes reasoning and proving capabilities from assurance properties, helping readers compare reported evidence and identify gaps relevant to their intended applications.

1.1 Objectives and Scope

The objectives of this survey are to identify the PL reasoning and proving tasks supported by NeSy frameworks and distinguish these capabilities from their reported assurance properties and mechanisms. The survey also examines how both are represented across existing surveys and primary publications. A further objective is to provide reusable annotations and a reproducible evaluation combining qualitative analysis and quantitative assessment that researchers can inspect and extend.

This survey focuses on PL to compare frameworks within a common logical setting while covering tasks such as satisfiability decision, witness generation, logical consequence, resolution, proof generation and validation, and neural guidance of symbolic solvers. Predicate, temporal, and other logics introduce additional constructs and reasoning requirements. Extending the classification to these logics would therefore require further taxonomy development. The chronological discussion contains 181 publications identified within the 1940–2026 search period. Of these, 180 are quantitatively evaluated. The work of McCulloch and Pitts (1943) is retained for historical context and excluded from the canonical evaluation dataset.

1.2 Contributions

The survey makes the following contributions:

1. A common conceptual foundation for PL reasoning and theorem proving, ANN, NeSy reasoning, and the SKOS ontology Miles and Bechhofer. It clarifies distinctions among inference, reasoning procedures, proving,

satisfiability decisions, satisfying witnesses, entailment, and formal derivations, enabling readers from neural and symbolic backgrounds to interpret the surveyed frameworks and taxonomy consistently.

2. An X–Y–Z taxonomy that separates reported capabilities from supporting assurance mechanisms and derives quantitative summaries from their annotations. Its implementation as an instance of SKOS ontology provides explicit definitions and reusable identifiers without treating the hierarchy as a ranking of NeSy systems.
3. An analysis of 26 existing NeSy survey and review papers, aligning their classifications with the *X* and *Y* dimensions. This comparison distinguishes the coverage of assurance in survey methodologies from the capabilities reported in the primary studies they discuss.
4. A chronological synthesis of 181 publications, with publication-level X–Y–Z annotations and ANN architecture labels. The accompanying discussions distinguish reasoning and proving tasks in PL, formula representations, and reported evidence, providing a basis for examining developments without defining periods by a presumed dominant architecture.
5. A formally defined evaluation function and an automated Python pipeline operating on curated publication and concept-assignment tables. Capability and assurance scores combine concept-frequency weights with coverage, while joint coverage summarizes the breadth of both dimensions. The pipeline computes concept and ANN architecture frequencies and yearly means, medians, and binned Shannon entropy, alongside publication counts. Annotation requires examination of the publications, whereas automation applies to the subsequent calculations.
6. An evidence-based analysis of underrepresented assurance characteristics and the limitations of the quantitative results. The released SKOS taxonomy, input data, Python implementation, search prompt, and SPARQL query with its results support inspection, reuse, and updates to the analysis.

1.3 Novelty and Main Findings

The novelty of this synthesis lies in connecting a PL-focused capability and assurance taxonomy with an analysis of existing survey classifications, publication-level annotations, and reproducible quantitative summaries. The evaluation describes the reported evidence in the corpus: taxonomy identifiers are not ordinal levels, concept frequency is not scientific importance, and a higher score does not by itself demonstrate better task performance or stronger guarantees. The findings indicate an imbalance

between commonly reported SAT-related capabilities and less frequently documented proving and assurance mechanisms. As reported in Table 33 in Section 6, SAT solving is assigned to 117 of the 180 evaluated publications, satisfiability witness generation to 69, and proof validation to 2. Regarding the ANN architectures used in these publications, Table 34 shows that the HNN label is assigned to 42 publications, DHNN to 31, and GNN to 27 publications. Section 6.5 further shows that positive yearly capability means coexist with zero assurance medians in nearly every evaluated year. These observations motivate greater attention to proof-producing methods, validation, and explicit assurance reporting. They do not establish that unannotated frameworks are unsound or that the wider field has uniformly declined. Missing annotations may reflect reporting limitations, annotation choices, or incomplete literature coverage. Researchers should therefore distinguish empirical success, understandable explanations, and formally supported conclusions when assessing their own and others' work, and state what evidence supports each claim.

1.4 Overview of the Survey

Section 2 introduces PL, distinguishes inference, reasoning, and proving, and supplies the ANN and SKOS background needed for the survey. Section 3 defines the X-Y-Z taxonomy for annotating surveyed papers according to the capabilities and assurance properties of the NeSy frameworks they describe, and explains the associated evaluation scores. Section 4 provides an overview of existing survey and review papers on NeSy AI, and annotates these papers using the capability and assurance concepts of the proposed X-Y-Z taxonomy, with a focus on reasoning and proving in PL. Section 5 presents the chronological literature survey and its publication-level annotation tables. Section 6 defines the evaluation function, describes data preparation, and analyzes concept frequencies, neural architectures, and annual score distributions. Section 7 summarizes the findings. Appendix A provides links to the GitLab implementation, SKOS taxonomy, and supplementary queries, together with the search prompt used to identify candidate publications, allowing readers to inspect and extend the evaluation independently.

2 Background

The background material provides readers from neural, symbolic, and interdisciplinary fields with a common foundation for understanding the survey. Basic AI knowledge is assumed, but advanced ANN expertise and prior familiarity with SKOS are not required. PL is covered in greater detail, while ANN concepts are introduced as needed. SKOS concepts and relations are explained to support understanding of the taxonomy.

2.1 Propositional Logic

In this section, PL is introduced before addressing its inference, reasoning, and proving tasks. PL is based on propositions. Not all sentences express propositions. For example, questions or commands are excluded. The alphabet of PL consists of four pairwise disjoint sets

$$\Sigma = P \cup M \cup 0 \cup C. \quad (1)$$

The set P is defined as a countably infinite set of atomic propositional letters, also called atomic words [Gorré \(2006\)](#), simple or atomic propositions ([Al-Odhari 2021](#), p. 36, Definition 2.2), propositional variables ([Zhang and Zhang 2022](#), p. 30, Section 2.1), written as $p_1, p_2, p_3, p_4, \dots, p_n$, etc. The set M of punctuation marks contains the parentheses "(" and ")". The set 0 includes the logical connective symbols, including conjunction ($\wedge$), disjunction ($\vee$), negation ($\neg$), implication ($\rightarrow$), and equivalence ($\leftrightarrow$) [Gorré \(2006\)](#). The set C of logical constants comprises the symbols $\top$ and $\perp$.

Every propositional variable $p_i \in P, i = 1, \dots, n$, as well as the constants $\top$ and $\perp$, is a well-formed formula. If p_1 is a w.f.f., then its negation $\neg p_1$ is also a w.f.f. A literal is defined as a propositional variable or its negation ([Zhang and Zhang 2022](#), p. 99, Section 3.3.1). Similarly, if p_1 and p_2 are w.f.f.s, then the formulas $(p_1 \vee p_2)$, $(p_1 \wedge p_2)$, $(p_1 \rightarrow p_2)$, and $(p_1 \leftrightarrow p_2)$ are w.f.f.s as well [Al-Odhari \(2021\)](#). Only the aforementioned rules are used to construct the w.f.f. in PL. As an illustration, let p_1 denote the sentence "it is raining" and let p_2 denote the sentence "the umbrella is open". Then the formula $(p_1 \rightarrow p_2)$ is a w.f.f., expressing "if it is raining, then the umbrella is open". As another example, the PL formula $(p_1 \wedge \top)$ is a w.f.f., whereas the PL expression $(p_1 \vee \top \vee)$ is not, since it does not follow rules for constructing w.f.f.s in PL [Al-Odhari \(2021\)](#).

A truth assignment is a function $I : P \rightarrow \{T, F\}$ that assigns a truth value to each propositional variable ([Poole and Mackworth 2010](#), Section 5.1.2). This assignment extends uniquely to all well-formed formulas using the truth conditions of the logical connectives [Al-Odhari \(2021\)](#). Formally, this interpretation (also called a valuation) is precise and rigorously specified ([Al-Odhari 2021](#), p. 37, Definition 2.1.1). Different interpretations may assign different truth values to the same propositional variable [Gorré \(2006\)](#). The interpretation function I assigns truth values to any w.f.f. in PL ([Enderton 2001](#), p. 20, Section 1.2). Truth tables provide a systematic method for determining the truth value of PL statements. Let p_1 and p_2 be atomic propositions. Then the interpretation function assigns true to $\neg p_1$ if and only if (iff) it assigns false to p_1 . For $p_1 \wedge p_2$, the interpretation function assigns true only when both p_1 and p_2 are true; otherwise the formula is false.

Similarly, for $p_1 \vee p_2$, the interpretation function assigns true iff at least one of them is true, and false otherwise. The proposition $p_1 \rightarrow p_2$ is assigned false only when p_1 is true and p_2 is false. In all other cases, it is true. Finally, the proposition $p_1 \leftrightarrow p_2$ is assigned true iff p_1 and p_2 have the same truth value, and false otherwise. In addition to truth value semantics, PL also admits an algebraic semantics [Avigad \(2022\)](#), which is not covered in this work.

Any w.f.f. in PL can be rewritten in either CNF or DNF. A CNF formula is a conjunction of disjunctions of literals (see Formula 2), whereas a DNF formula is a disjunction of conjunctions of literals (see Formula 3) ([Liberatore 2008](#), Section 3).

$$(p_1 \vee p_2) \wedge (\neg p_3 \vee p_4) \quad (2)$$

$$(p_1 \wedge \neg p_3) \vee (p_1 \wedge p_4) \vee (p_2 \wedge \neg p_3) \vee (p_2 \wedge p_4) \quad (3)$$

A PL clause is a disjunction of literals. A Horn clause in PL is a special form of CNF containing at most one positive literal ([Liberatore 2008](#), p. 267, Section 3). For example, formula (4) is a Horn clause.

$$(\neg p_1 \vee p_2 \vee \neg p_3) \quad (4)$$

A Horn formula is a conjunction of Horn clauses ([Liberatore 2008](#), p. 268, Section 3).

In the following three sections, 2.2, 2.3, and 2.4, inference, reasoning, and proving in PL are described. Understanding these three concepts provides a foundation for the discussions in the rest of this survey. The distinctions among inference, reasoning, and proving are adopted as working definitions for organizing this survey, rather than as universally standardized or mutually exclusive categories, acknowledging that terms such as inference procedure, reasoning algorithm, and proof procedure have overlapping and context-dependent uses in logic and AI. Figure 1 shows a Venn diagram which illustrates inference, reasoning, and proving approaches in PL. Section 2.6 describes AI reasoning paradigms referenced in the remaining part of this survey.

2.2 Inference in Propositional Logic

In PL, an inference denotes a formal mechanism for deriving a PL statement, called the conclusion, from one or more PL statements, called the premises. Such a mechanism is specified by an inference rule, which defines a form of derivation ([Enderton 2001](#), p. 110). An inference rule is said to be truth-preserving or valid if, whenever all premises are true, the conclusion must

also be true. An example of an inference rule is Modus ponendo ponens (MPP) [Schechter and Enoch \(2006\)](#), which is formally expressed as

$$\frac{P, \quad P \rightarrow Q}{Q} \quad (5)$$

where P, Q are atomic propositions. It states that if the atomic proposition P is true ($\top$), and the implication $P \rightarrow Q$ is also true, then the atomic proposition Q must be true. This property can be verified using truth tables. In contrast, the argument form (6) is not a valid inference rule in PL because it is not truth-preserving. In particular, there exists a truth assignment to P for which the premise is true while the conclusion is false.

$$\frac{P}{\perp} \quad (6)$$

It is important to note that the validity of an inference rule depends on the rule itself, not on the truth values assigned to the atomic propositions. Even if one or more of the premises is false, denoted by $\perp$, the conclusion may be either true or false. However, the inference rule itself remains truth-preserving. A **resolution** step (hereafter simply resolution) in PL is another fundamental inference rule in PL ([Zhang and Zhang 2022](#), p. 98–99, Section 3.3.1). It can be expressed as follows:

$$\frac{(P \vee Q), (\neg P \vee R)}{(Q \vee R)} \quad (7)$$

where P, Q , and R are atomic propositions. In resolution, it is allowed to use negation of atomic propositions, such as $\neg P$. The resolution rule is truth-preserving, meaning that no truth assignment to atomic propositions can be found such that the premises are true and the conclusion is false.

2.3 Reasoning in Propositional Logic

The preceding section delineates inference in PL, namely, the utilization of inference rules such as MPP and resolution. It is a rule-based mechanism that derives a conclusion from given premises. By contrast, reasoning in PL refers to an algorithmic process that applies fixed, truth-preserving inference rules. This process relies on additional mechanisms such as the ordering of rule applications, termination criteria, search strategies, and redundancy elimination.

Truth tables in PL are not reasoning methods in themselves. Rather, they provide a (semantic) procedure for evaluating the truth values of **PL** formulas under all possible interpretations. By examining a truth table, one can determine fundamental logical properties such as satisfiability (SAT), tautology (validity), and logical entailment.

Entailment in PL, written as $p_1 \models p_2$, is a semantic property of PL. It means that in every interpretation in which p_1 is true, p_2 is also true. **Equivalently**, p_2 is a logical consequence of p_1 . Under entailment, there is no truth assignment under which p_1 is true and p_2 is false.

A PL expression (8) represents one w.f.f. **Exhaustive truth tables** can serve as a semantic decision procedure for determining which assignments make this w.f.f. true.

$$(p_1 \vee p_2) \wedge \neg p_1 \quad (8)$$

The truth value of formula (8) depends on the truth values assigned to the propositional variables p_1 and p_2 . As shown in Table 2, the interpretation function assigns true to w.f.f. (8) when p_1 is false and p_2 is true. This demonstrates that w.f.f. (8) is satisfiable [Gorré \(2006\)](#), since there exists a truth value assignment to the propositional variable that makes the formula true.

Table 2. Truth-value assignments of $(p_1 \vee p_2) \wedge \neg p_1$.

p_1	p_2	$\neg p_1$	$p_1 \vee p_2$	$(p_1 \vee p_2) \wedge \neg p_1$
T	T	F	T	F
T	F	F	T	F
F	T	T	T	T
F	F	T	F	F

There are several variants of SAT problems, including MaxSAT ([Marino 2021](#), Section 2), 3SAT or 2SAT ([Kho et al. 2022](#), Section 2.1). The MaxSAT problem involves finding a truth value assignment to propositional variables that maximizes the number of satisfied clauses in a CNF formula. In other words, a clause is considered satisfied if at least one of its **literals** is true ([Marino 2021](#), Section 2). In the 3SAT problem, we are given a set of propositional variables and a collection of clauses, where each clause contains exactly three literals. The 3SAT task is to determine whether there exists a truth assignment for the finite set of propositional variables that satisfies all clauses in the collection ([Garey and Johnson 1979](#), p. 46, Section 3.1). A similar definition applies to the 2SAT problem in PL ([Kho et al. 2022](#), Section 2.1).

A **resolution** method is a syntactic reasoning technique in PL ([Harrison 2009](#), p. 179, Section 3.11). By contrast, the tableau method is a semantic reasoning technique in PL that provides a decision procedure for the SAT problem ([Jarmužek 2020](#), p. 29, Section 2.3, Definition 2.10). It incrementally constructs a tree structure for a PL formula by systematically decomposing it

into simpler components using tableau rules (Jarmużek 2020, p. 29, Section 2.3, Definition 2.13).

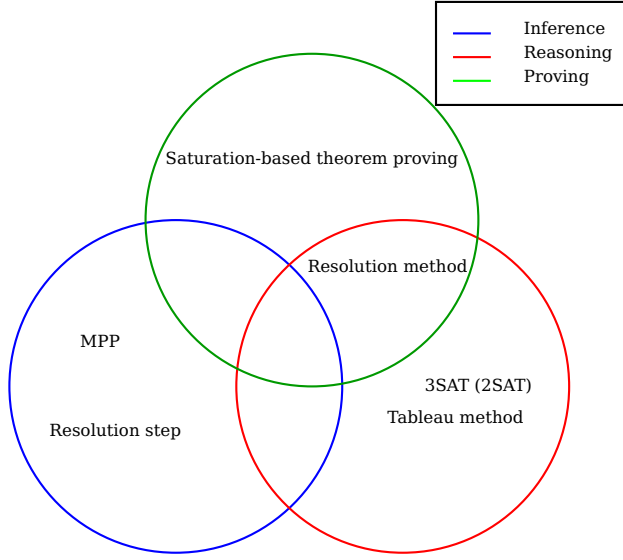


Figure 1. Venn diagram illustrating inference, reasoning, and proving approaches in PL.

In PL, a w.f.f. is a tautology if it is true under every possible truth assignment to its atomic propositions (Enderton 2001, p. 23). Consequently, a w.f.f. is a contradiction if it is false under every possible truth assignment to its atomic propositions.

For satisfiability in PL, a witness is a truth assignment that makes the formula true. For example, assigning both p and q the value true is a satisfiability witness for $p \wedge q$. Witness generation is the task of finding such an assignment.

A lemma is a proved statement used to help establish another result. For example, a statement derived during proof search may be retained as a lemma and reused in later steps.

2.4 Theorem Proving in Propositional Logic

In classical PL, let Γ be a finite set of propositional formulas and let φ be a propositional formula; the premises and conclusion need not be atomic. In a specified proof calculus, $\Gamma \vdash \varphi$ means that there is a finite formal derivation of φ from Γ , using the axioms and inference rules of that calculus. By contrast, $\Gamma \models \varphi$ means that every truth assignment satisfying all formulas

in Γ also satisfies φ . For a sound and complete proof calculus for classical PL, $\Gamma \vdash \varphi$ if and only if $\Gamma \models \varphi$; soundness gives the forward implication and completeness the reverse Plaisted (2014). A theorem in PL is a formula provable without undischarged premises, written $\vdash \varphi$. The term *theorem proving* is also commonly used for constructing proofs from premises, thereby establishing $\Gamma \vdash \varphi$ and certifying $\Gamma \models \varphi$. For finite Γ , this can equivalently be formulated as proving $(\bigwedge_{\gamma \in \Gamma} \gamma) \rightarrow \varphi$ without premises, with the empty conjunction interpreted as the truth constant $\top$.

The entailment $\Gamma \models \varphi$ holds exactly when $\Gamma \cup \{\neg\varphi\}$ is UNSAT, meaning that no truth assignment satisfies every formula in this set. If this set is satisfiable, any satisfying assignment is a countermodel to the entailment: it makes all premises true and the conclusion false.

With no premises, $\vdash \varphi$ is equivalent to $\models \varphi$, meaning that φ is a tautology, or valid formula: it is true under every truth assignment. Equivalently, $\neg\varphi$ is unsatisfiable Plaisted (2014).

A clause-based proof from an initial clause set S can be represented as a finite sequence of clauses, each of which belongs to S or follows from earlier clauses by an inference rule. The final clause is the derived conclusion. In PL, the resolution rule derives $C \vee D$ from $C \vee p$ and $D \vee \neg p$, where C and D are disjunctions of literals, possibly empty. For example, resolution derives q from the clauses p and $\neg p \vee q$. A derivation of the empty clause $\square$, which represents falsehood, is called a resolution refutation. To establish $\Gamma \models \varphi$ by refutation, one converts $\Gamma \cup \{\neg\varphi\}$ into an equisatisfiable finite clause set S and seeks a derivation of $\square$ from S . Equisatisfiability means that S is satisfiable if and only if the original set is satisfiable. By soundness, a refutation establishes that S is unsatisfiable. Equisatisfiability then implies that $\Gamma \cup \{\neg\varphi\}$ is unsatisfiable and hence that $\Gamma \models \varphi$.

For example, to establish $\{p, p \rightarrow q\} \models q$, one refutes $S = \{p, \neg p \vee q, \neg q\}$ by deriving q and then resolving it with $\neg q$ to obtain $\square$. Resolution in PL is refutationally complete iff every UNSAT finite clause set has a resolution refutation Bachmair and Ganzinger (2001). This property suffices for proving entailments by refutation, but it does not imply that resolution alone can directly derive every logically implied clause.

Resolution-based proof search can be organized as a saturation process, which repeatedly applies inference rules to seek a clause set closed under those rules. Search can stop earlier if the empty clause is derived. More general saturation procedures consider closure modulo a redundancy criterion that preserves completeness Bachmair and Ganzinger (2001).

2.5 A Short Introduction to *Artificial* Neural Networks

An ANN serves as the neural component in a NeSy framework and can be viewed as a simplified model of the networks of neurons in the human brain Gurney (1997). A simple artificial neuron is illustrated in (Gurney 1997, Figure 1.2). In ANNs, synapses are represented by single numerical weights that multiply each input before it reaches the neuron’s equivalent of the cell body. The weighted inputs are then summed using arithmetic addition. One of the simplest models of an artificial neuron is the Threshold Logic Unit (TLU) (Gurney 1997, Section 1.1), which computes the weighted sum of its inputs and generates an output based on a threshold. A network of such artificial neurons forms an ANN Gurney (1997).

ANNs underpin many modern AI systems. ConvNNs process grid-structured data, such as images, using shared learnable filters to extract local features and combine them into more complex representations (LeCun et al. 1998). GNNs process graph-structured data by iteratively combining information from neighboring nodes and their connections to learn representations for node-level or graph-level predictions (Scarselli et al. 2009).

2.6 NeSy AI Reasoning

Reasoning paradigms are described with particular emphasis on NeSy AI reasoning, as presented in the survey Liang et al. (2025). These paradigms are referenced throughout the remainder of this paper. Figure 2 illustrates the categorization of AI reasoning proposed in that survey (Liang et al. 2025, Section 2.2.2).

NeSy AI reasoning integrates an ANN with symbolic representations and reasoning mechanisms. As described in the survey (Liang et al. 2025, Section 2.2.1, formula 6), NeSy reasoning can be formalized as a function with two inputs: an observation represented as a query or graph and encoded by a neural encoder, and knowledge expressed as rules or graphs. In the case of PL, rules can be expressed either as PL implications or in CNF. The reasoning output is then derived jointly from both the neural and symbolic components.

Deductive, inductive, and abductive reasoning are classified under the task structure category in Liang’s taxonomy (see Figure 2). Deductive reasoning derives logically valid conclusions from known premises or rules (Liang et al. 2025, Section 2.2.2). In contrast, inductive reasoning generalizes from specific instances to broader rules or models (Liang et al. 2025, Section 2.2.2). In both cases, premises, rules, or instances can be expressed in PL, typically using implications. Abductive reasoning, on the other hand, seeks the most plausible explanation for a given observation (Liang et al. 2025,

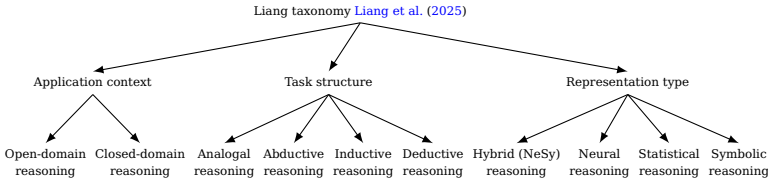


Figure 2. Tree-shaped taxonomy for categorizing AI reasoning with a focus on NeSy AI [Liang et al. \(2025\)](#)

Section 2.2.2). Its formalization in PL is provided in ([Rasga and Sernadas 2022](#), p. 705, Definition 3.2).

Neural reasoning is classified under the representation type of reasoning in Liang’s taxonomy (see Figure 2). It relies on learned representations within an ANN. Unlike formal logic reasoning, neural reasoning does not use explicit rule structures. The inference outcome is predicted using multi-layer transformations and pattern abstraction ([Liang et al. 2025](#), Section 2.2.2). Essentially, it performs pattern-based inference rather than reasoning in the traditional formal logic sense. In some approaches, neural reasoning can also leverage graph-based embeddings of PL formulas [Ghanem \(2024\)](#).

Hybrid reasoning in NeSy AI falls under the representation type category in Liang’s taxonomy (see Figure 2) [Liang et al. \(2025\)](#). It leverages the flexibility to unify the interpretability of formal logic with the learning capabilities of ANN. As a result, hybrid reasoning performed by NeSy frameworks can be understood and explained in human-readable terms.

In the literature on NeSy AI, the terms inference and reasoning are used interchangeably. For example, in [Evans and Grefenstette \(2018\)](#), the authors use the term inference to describe a specific algorithm, named forward and backward chaining inference, which implements logical reasoning over Horn clauses expressed in first order logic. In the standard AI literature, such as in [Russell and Norvig \(2010\)](#), forward chaining is explicitly described as a reasoning algorithm for knowledge bases written in Horn form ([Russell and Norvig 2010](#), p. 275). These two naming conventions do not conflict. Reasoning refers to the abstract logical process of deriving conclusions [Russell and Norvig \(2010\)](#), whereas inference executes this reasoning within a NeSy system [Evans and Grefenstette \(2018\)](#).

2.7 SKOS: A Simple Knowledge Organization System

Before reading Section 3, it is necessary to understand the basic notions of ontologies and SKOS, briefly introduced below.

A formal ontology represents knowledge about a domain using logical statements called axioms ([Hitzler et al. 2012](#), Sections 2–3). In an

ontology, classes represent categories of things, individuals represent particular objects, and object and data properties describe relationships between individuals and associations between individuals and data values, respectively. The term *formal* means that ontology statements have precise, mathematically defined semantics, which specify their logical consequences and provide the basis for automated reasoning, including consistency checking (Hitzler et al. 2012, Sections 2–3).

SKOS is a standard for representing and sharing organized vocabularies, including taxonomies Miles and Bechhofer (2009). The SKOS data model is formally defined as an ontology Miles and Bechhofer. It represents ideas as concepts, which can have labels, definitions, and relationships to other concepts. Concepts can be organized into concept schemes. The prefix `skos:` abbreviates the SKOS namespace. The class `skos:Concept` represents SKOS concepts, while `skos:prefLabel` provides a preferred lexical label, with at most one such label per language tag. The object properties `skos:broader` and `skos:narrower` link a concept to a broader and a narrower concept, respectively (Miles and Bechhofer 2009, Sections 1.2, 2–5, 7–8). For example, stating that `cat` is a narrower concept than `animal` does not by itself entail the logical statement *every cat is an animal* (Miles and Bechhofer 2009, Section 1.3). Here, `skos:broader` and `skos:narrower` organize the taxonomy presented in Section 3.

SKOS was selected because it is a W3C-standardized model for representing taxonomies and supports the concepts, labels, definitions, and hierarchical relationships required by the classification (Miles and Bechhofer 2009, Sections 1.1–1.3). Its RDF-based representation facilitates sharing and reuse, while providing sufficient expressiveness for organizing and annotating NeSy capabilities and assurance mechanisms.

3 A Taxonomy for Surveying NeSy Frameworks

The X–Y–Z taxonomy supports the annotation of surveyed papers according to the reported capabilities and assurance properties of the NeSy frameworks they describe. One framework is analyzed per paper, with each paper serving as the unit of annotation and evaluation. The taxonomy comprises three dimensions:

The *X* dimension describes the reasoning tasks (e.g., SAT decision), mechanisms, and capability-related properties that NeSy frameworks support or exhibit in PL.

The *Y* dimension describes assurance mechanisms and properties that support confidence in a NeSy framework’s reasoning, results, and correctness claims.

The Z dimension provides metrics computed from the assigned X and Y annotations to summarize reported capability and assurance profiles for quantitative evaluation. It does not define a separate classification hierarchy.

An X concept is assigned when the publication provides evidence of the corresponding capability or characteristic. A Y concept is assigned when the publication provides evidence of the corresponding assurance mechanism or property. Assigning a Y concept does not imply that the framework satisfies all assurance criteria.

Related X and Y concepts may have similar names, but an X annotation does not automatically imply a corresponding Y annotation. Together, these dimensions provide a structured basis for the systematic comparison of NeSy frameworks described in surveyed papers and for identifying limitations in existing approaches. Within the X and Y hierarchies, numerical codes identify concepts and their positions in the broader–narrower structure. They do not indicate rankings of capability, sophistication, or performance, or progression between categories. A child concept is more specific than its parent.

The category-level coding tables below support reproducible annotation by specifying necessary evidence, sufficient coding evidence, exclusion criteria, borderline-case decisions, and examples. The guidance clarifies parent–child annotation and distinguishes soundness guarantee ($X7$) from soundness assessment ($Y3$), and interpretability ($X8.2$) from interactive explanation ($Y2$). Direct parent annotation requires operational evidence establishing the parent concept’s scope without supporting a listed child for the same claim. Parent concepts are not automatically assigned when a child is annotated. Redundant assignments and assignments based on vague evidence are excluded. Parent and child codes may coexist only for separately evidenced claims or scopes. Parent examples must correspond to direct annotations and include locators for the supporting evidence. Parents without qualifying direct annotations are identified as organizational or unobserved in the corpus.

3.1 How to Read the X – Y – Z Taxonomy?

The X – Y – Z taxonomy should be read as a SKOS concept scheme rather than a formal logical class hierarchy with disjointness axioms. The X dimension describes the reasoning tasks, mechanisms, and capability-related properties that a NeSy framework supports or exhibits in PL, such as SAT decision, constructive evidence generation, solution strategy, search strategy, learning robustness, architectural capability, soundness guarantee, and reliability. The Y dimension describes the evidence and procedures used to assess or substantiate the correctness claims associated with these capabilities, including verification, certification, proof validation,

explainability, or soundness checking. Thus, related X and Y concepts may have similar names, but they play different roles: X captures capability, while Y captures the assurance provided for that capability. The Z dimension provides metrics for the subsequent quantitative evaluation of surveyed frameworks rather than a separate hierarchy of classification concepts. The SKOS implementation of the taxonomy is available as a TTL file at the URL provided in Listing 3 of Appendix A.1.

3.2 X : NeSy Reasoning and Proving Capability Level in PL

Figure 3 illustrates the X dimension of the taxonomy, which characterizes the **capability level (X0)** of a NeSy framework **for reasoning and proving tasks in PL**. This dimension records the reasoning and proving tasks, mechanisms, and capability-related properties reported for a NeSy framework in PL. The X dimension is organized into eight groups: **logical foundation (X1)**, **constructive evidence generation (X2)**, **solution strategy (X3)**, **search strategy (X4)**, **learning robustness (X5)**, **architectural capability (X6)**, **soundness guarantee (X7)**, and **reliability (X8)**.

3.2.1 Foundational Reasoning Logical foundation (X1) covers satisfiability decision (X1.1) and logical consequence (X1.2) in classical PL. A model, in the logical sense, is a Boolean truth assignment satisfying a formula F . The formula F is SAT if a model exists and UNSAT otherwise. Logical consequence, $\Gamma \models \varphi$, means that every model of the premises Γ satisfies φ (Russell and Norvig 2003, p. 203); equivalently, $\Gamma \cup \{\neg\varphi\}$ is unsatisfiable. The annotation (X1.1) records an explicitly reported SAT/UNSAT decision task, not evidence generation or a correctness guarantee.

Table 3 specifies the necessary and sufficient coding evidence, exclusion criteria, borderline cases, and candidate examples for $X1$, $X1.1$ and $X1.2$ concepts.

Table 3. Coding criteria for $X1$, $X1.1$, $X1.2$ concepts.

Code	Necessary evidence	Sufficient coding evidence	Exclusion criteria	Typical borderline case and decision	Candidate example
$X1$	An explicitly specified classical-PL reasoning task with operational support.	A task definition and procedure establish reasoning over classical propositional formulas.	"Logical" terminology alone; symbolic processing without an established classical-PL reasoning task.	CNF syntax alone is insufficient; the publication must establish the associated classical-PL reasoning task.	×
$X1.1$	An explicit SAT/UNSAT decision task over PL formulas.	A described decision procedure or predictor producing SAT/UNSAT outputs.	Generic classification without a stated satisfiability task.	An imperfect SAT classifier qualifies for the task code, but not for a soundness guarantee.	Kibria (2007)
$X1.2$	An explicit entailment or consequence task with premises and conclusions.	A reported procedure that predicts or derives whether $\Gamma \models \varphi$.	Merely noting that entailment can be reduced to SAT.	A SAT-based entailment procedure qualifies only when the entailment task is explicitly reported.	França et al. (2014)

- X0: Capability level
 - X1: Logical foundation
 - X1.1: Satisfiability decision
 - X1.2: Logical consequence
 - X2: Constructive evidence generation
 - X2.1: Satisfiability witness generation
 - X2.2: UNSAT refutation
 - X3: Solution strategy
 - X3.1: Logical strategy
 - X3.1.1: Solution method
 - X3.1.1.1: SAT solving
 - X3.1.1.2: Resolution
 - X3.1.2: Heuristic adaptation
 - X3.1.2.1: Strategy monitoring
 - X3.1.2.2: RL-guided clause selection
 - X3.1.2.3: Query-aware formula generation
 - X3.1.2.4: Strategy adaptation
 - X3.2: MaxSAT optimization
 - X4: Search strategy
 - X4.1: Lemma discovery
 - X4.2: Clause selection quality
 - X4.3: Search-space pruning
 - X4.4: Branching heuristic
 - X4.5: Heuristic optimization for SAT
 - X4.6: Lemma reuse
 - X5: Learning robustness
 - X5.1: Logical generalization
 - X5.2: Paradox-resistant training
 - X5.3: UNSAT-core prediction
 - X6: Architectural capability
 - X6.1: Algorithmic alignment
 - X6.2: Logic expressivity
 - X7: Soundness guarantee
 - X8: Reliability
 - X8.1: Robustness
 - X8.2: Interpretability

Figure 3. X dimension: taxonomy for NeSy reasoning and proving capability level

3.2.2 *Constructive Evidence Generation* Constructive evidence generation (X2) concerns explicit evidence for SAT or UNSAT, rather than a decision label alone. Satisfiability witness generation (X2.1) produces a model of the input formula; a predicted assignment or an energy minimum is insufficient unless the publication establishes that it satisfies the formula. UNSAT refutation (X2.2) concerns solver-based construction of a refutation: a formal

demonstration that the input has no model. In a clause-based proof system, a refutation typically ends with the empty clause, which is false under every assignment. A bare UNSAT prediction is not a refutation. The reported mechanisms for checking such evidence are annotated separately under *Y*.

Table 4 specifies the evidence required to code *X2*, *X2.1* and *X2.2*, alongside exclusion criteria, borderline cases, and a candidate example, distinguishing constructive evidence from unsupported predictions.

Table 4. Coding criteria for *X2*, *X2.1*, *X2.2* concepts.

Code	Necessary evidence	Sufficient coding evidence	Exclusion criteria	Typical borderline case and decision	Candidate example
<i>X2</i>	A mechanism constructing explicit evidence associated with a reported PL reasoning result.	A documented construction procedure, supported by an example or formal argument, establishes the output's evidential role for the result.	Bare SAT/UNSAT labels; confidence scores; checking existing objects without constructing evidence.	A candidate output lacking an established evidential role does not justify <i>X2</i> simply because no child can be confirmed.	×
<i>X2.1</i>	An assignment associated with the input formula and evidence of satisfaction.	A model-producing procedure supported by reported satisfaction checking or a validity argument.	Unchecked candidate assignments, energy minima, or confidence scores.	A candidate checked to satisfy its input qualifies for the demonstrated scope, but not as a universal guarantee.	Xie et al. (2024)
<i>X2.2</i>	Construction of a formal refutation of the input.	A specified proof-producing mechanism with a refutation example or correctness argument.	Bare UNSAT labels, predicted cores, or validation-only systems.	Invoking a solver that returns UNSAT does not qualify unless refutation construction is documented.	Sun et al. (2026)

3.2.3 Solution Strategy Solution strategy (*X3*) encompasses approaches used by NeSy frameworks to organize and carry out reasoning and proving tasks in PL. This concept includes logical strategy (*X3.1*) and MaxSAT optimization (*X3.2*). Logical strategy comprises solution method (*X3.1.1*) and heuristic adaptation (*X3.1.2*).

In addition, heuristic adaptation provides more flexible reasoning approaches, including strategy monitoring (*X3.1.2.1*), RL-guided clause selection (*X3.1.2.2*), query-aware formula generation (*X3.1.2.3*), strategy adaptation (*X3.1.2.4*).

MaxSAT optimization (*X3.2*) denotes a NeSy framework’s ability to solve a MaxSAT problem as part of its reasoning process with the objective of finding a truth assignment that maximizes the number or total weight of satisfied clauses Zhang et al. (2024a).

Table 5 specifies coding criteria for *X3* concepts, covering necessary evidence, sufficient coding evidence, exclusion criteria, typical borderline cases and decisions, and candidate example papers.

SAT solving (*X3.1.1.1*) covers SAT-oriented prediction or search. The accompanying annotation (*X3.1.1.1*) record specifies whether the learned component predicts a decision, searches for assignments, guides a symbolic solver, or combines these roles. Resolution (*X3.1.1.2*) is a clause-inference

Table 5. Coding criteria for *X3* concepts.

Code	Necessary evidence	Sufficient coding evidence	Exclusion criteria	Typical borderline case and decision	Candidate example
X3	An explicit approach organizing how a PL reasoning problem is solved.	A documented workflow establishes the roles and coordination of the reasoning or solving components.	Method names, component inventories, or performance comparisons without an operational solution account.	A hybrid label is insufficient; documented component roles and coordination can establish the parent scope.	×
X3.1	A logic-based reasoning or solution-control approach with operational support.	Logical operations or control principles are described sufficiently to establish how the approach functions.	Processing PL encodings without a documented logic-based strategy; generic optimization alone.	A learned controller's use in a logic solver is insufficient unless its logic-based control role is established.	×
X3.1.1	A concrete logical solution method supported by operational or formal details.	Specified procedural steps or a formal definition establish how the logical method solves the stated PL problem.	Unexplained method names; black-box solver calls; heuristic tuning without an established logical solution method.	An unexplained method name is insufficient; a concrete procedure may support the parent when no listed subtype is evidenced.	×
X3.1.1.1	An explicitly described SAT-oriented computational method, with the role of its neural component identified.	A documented method in which the neural component predicts SAT/UNSAT, searches for candidate assignments, or guides a symbolic SAT solver.	Merely naming SAT as an application domain or reporting a neural component without specifying its SAT-related computational role.	Include the neural role in the annotation record: prediction, assignment search, solver guidance, or a documented combination. The concept identifier alone does not distinguish these roles or establish witness validity, refutation generation, or soundness.	Delacour et al. (2025)
X3.1.1.2	Resolution inference is part of the framework's reported computation.	An algorithm or trace showing resolution steps or documented invocation of a resolution-based method.	Mentioning resolution only as background.	Resolution-based reasoning need not export a refutation certificate.	Lima (2001)
X3.1.2	A mechanism for monitoring, learning, or adapting heuristic control.	A documented monitor, learned control rule, or adaptation loop establishes heuristic-control functionality.	Fixed heuristic choices without monitoring, learning, or adaptation; overall runtime reporting without heuristic-level evidence.	Monitoring alone may establish the scope; if a monitoring child is supported, code that child rather than the parent.	Ertel et al. (1989)
X3.1.2.1	Monitoring of reasoning/search state during operation.	A mechanism that tracks progress, state, or strategy-relevant indicators during solving.	Offline benchmark reporting alone.	Monitoring without strategy modification qualifies here, not automatically as adaptation.	×
X3.1.2.2	Reinforcement learning controls clause selection.	A reported state/action/reward or equivalent RL formulation whose actions select clauses.	Supervised ranking; RL for unrelated decisions.	RL selecting variables or tactics does not qualify unless those actions are actually clause selection.	Vaezipoor et al. (2020)
X3.1.2.3	A query conditions formula or clause generation/transformation.	A documented query-conditioned construction procedure.	Merely evaluating or querying an unchanged formula.	If queries guide search but do not generate formulas, rename the category rather than stretch the definition.	Ozolins et al. (2022)
X3.1.2.4	Strategy changes in response to the problem or intermediate state.	A described strategy-switching or parameter-adaptation rule with its triggering information.	A single fixed strategy.	A learned policy is not automatically adaptive merely because it was trained.	Zhang et al. (2024b)
X3.2	An explicit MaxSAT objective of maximizing the number or total weight of satisfied clauses.	An operationally described exact, approximate, or heuristic method that constructs or searches for assignments under the stated MaxSAT objective.	SAT classification, assignment prediction, or a generic SAT loss without an explicitly established MaxSAT optimization objective.	The maximum specifies the optimization objective, not a guaranteed outcome. Approximate and heuristic methods qualify; global optimality must be separately substantiated for the claimed scope.	Chen et al. (2025)

rule. Deriving the empty clause by resolution establishes UNSAT, not SAT. Failure to find a refutation within a search budget does not establish satisfiability. Strategy monitoring (X3.1.2.1) refers to the capability of a NeSy framework to dynamically monitor and enhance PL solution processes. Strategy adaptation (X3.1.2.4) refers to the capability of a NeSy framework to modify its reasoning or proving strategy in response to problem characteristics or intermediate results in order to improve its efficiency in PL reasoning and proving tasks. For example, as a form of strategy adaptation, the framework can modify its branching heuristic while determining the satisfiability of a PL formula.

RL-guided clause selection (X3.1.2.2) denotes the use of reinforcement learning to learn effective strategies for selecting clauses in NeSy theorem proving. In this setting, RL can identify novel and effective strategies for clause selection Suda (2025). Query-aware formula generation (X3.1.2.3) leverages the query mechanism within a NeSy framework to solve SAT problems in PL Ozolins et al. (2022).

3.2.4 Search Strategy Search strategy (X4) encompasses six key capabilities that strengthen NeSy reasoning and proving capability level: lemma discovery (X4.1), clause selection quality (X4.2), search-space pruning (X4.3), branching heuristics (X4.4), heuristic optimization for SAT (X4.5), and lemma reuse (X4.6) concepts.

Table 6 provides coding criteria for these six search-strategy concepts, specifying necessary evidence, sufficient coding evidence, exclusion criteria, typical borderline cases and decisions, and candidate example papers where available. These criteria support consistent, evidence-based annotation of surveyed papers.

Lemma discovery (X4.1) denotes the capability of a NeSy framework to derive useful auxiliary lemmas during proof search. Clause selection quality (X4.2) in the context of NeSy automated theorem proving in PL, refers to a framework’s ability to choose clauses during proof search in a way that minimizes the search space and reduces proof time Suda (2021). Search-space pruning (X4.3) addresses the SAT problem by eliminating unsatisfiable branches from the search space Heule et al. (2017). In NeSy frameworks, it refers to mechanisms that guide the exploration of the combinatorial search space during reasoning or proof in PL. Branching heuristics (X4.4) refer to an emerging heuristic approach used by NeSy frameworks to learn how to solve the propositional SAT problem Selsam et al. (2019).

Heuristic optimization for SAT (X4.5) refers to heuristic methods used by NeSy frameworks to solve SAT problems. These methods can combine global exploration and local improvement. Global exploration seeks promising assignments across different regions of the search space. Local improvement

Table 6. Coding criteria for *X4* concepts.

Code	Necessary evidence	Sufficient coding evidence	Exclusion criteria	Typical borderline case and decision	Candidate example
<i>X4</i>	A concrete mechanism controlling exploration of a reasoning or proof-search space.	A documented procedure specifies how search states, branches, or proof steps are selected, ordered, pruned, or scheduled.	Search terminology alone; training optimization alone; runtime improvements without an exploration-control mechanism.	A faster solver does not establish <i>X4</i> unless the mechanism controlling reasoning-space exploration is documented.	×
<i>X4.1</i>	Construction of auxiliary lemmas used in reasoning or proof search.	A reported lemma-generation mechanism with demonstrated or justified lemma validity.	Unsupported intermediate statements merely called lemmas.	Proposed lemmas require documented validation or a validity-preserving construction.	×
<i>X4.2</i>	Clause selection plus evidence concerning its quality.	Comparative empirical or analytical evidence of improvement in a named search-related measure.	Clause selection with no supporting quality evidence.	Better prediction accuracy alone is insufficient unless related to the stated selection-quality measure.	Moriyama et al. (2023)
<i>X4.3</i>	Removal of candidate states or branches from further search.	A described pruning rule or mechanism applied during reasoning.	Reordering or scoring candidates without excluding them.	Heuristic pruning may qualify without establishing completeness or soundness.	Feng et al. (2024)
<i>X4.4</i>	Selection of a branching variable, value, or polarity.	A documented decision rule used at search branch points.	Whole-assignment prediction or SAT classification alone.	A core predictor qualifies here only if its effect on branching is explicitly documented.	Farooque et al. (2024)
<i>X4.5</i>	Heuristic improvement or exploration of candidate SAT assignments.	A reported assignment-search procedure using local moves, global exploration, or both.	Merely describing a predictor as heuristic.	Branch-variable selection alone should not receive this code unless a distinct assignment-optimization mechanism is documented.	Chen et al. (2024)
<i>X4.6</i>	Previously established lemmas are retained and subsequently used.	A described storage, retrieval, and application mechanism.	Rediscovery without reuse; generic caching unrelated to lemmas.	Reusing lemmas within one run or across runs qualifies, but the scope is recorded.	×

modifies a current assignment through small changes, such as single-variable flips in GSAT Selman et al. (1992). These strategies are complementary rather than mutually exclusive. Lemma reuse (*X4.6*) refers to the ability of a NeSy framework to store previously established lemmas and apply them later during proof search.

3.2.5 Learning Robustness Learning robustness (*X5*) encompasses key concepts such as logical generalization (*X5.1*), paradox-resistant training (*X5.2*), UNSAT-core prediction (*X5.3*).

Table 7 specifies their necessary and sufficient coding evidence, exclusion criteria, borderline-case decisions, and candidate examples for *X5* concepts.

Specifically, logical generalization denotes the capability of a NeSy framework to apply learned logical patterns across domain-agnostic PL reasoning or proving tasks Chen (2025). Building on this, paradox-resistant training ensures that the reasoning mechanisms avoid generating paradoxes during training. UNSAT-core prediction (*X5.3*) uses learned predictions about core membership to guide a SAT solver (Selsam and Bjørner 2019, pp. 336–338). An UNSAT core is an unsatisfiable subset of the input clauses that

Table 7. Coding criteria for *X5* concepts.

Code	Necessary evidence	Sufficient coding evidence	Exclusion criteria	Typical borderline case and decision	Candidate example
<i>X5</i>	An explicit, supported learning-related robustness or generalization property with a stated scope.	Experiments or analysis support a defined robustness/generalization property within specified evaluation conditions.	Unqualified robustness claims; accuracy scores without a defined robustness/generalization evaluation; properties outside the learning-related scope.	Performance on another dataset qualifies only when the protocol supports a defined generalization claim; additional scores alone are insufficient.	×
<i>X5.1</i>	A defined generalization setting beyond the training distribution or problem regime.	Reported successful transfer or a formal generalization result in that setting.	Same-distribution held-out accuracy alone.	Transfer to larger formulas qualifies only when the train (test) size distinction and results are explicit.	França et al. (2014)
<i>X5.2</i>	A precisely defined targeted paradox/inconsistency-related training failure.	A training intervention plus evidence addressing that specified failure.	Ordinary UNSAT inputs; undefined claims of paradox resistance.	If paradox is not operationally defined, withhold the code and reconsider the category.	Abubakar et al. (2020b)
<i>X5.3</i>	Learned prediction of core membership or an UNSAT-core candidate.	A reported core-prediction mechanism used for solver guidance.	Treating predictions as established UNSAT evidence.	An approximate core predictor qualifies without producing a valid core or refutation.	Shi et al. (2023)

need not be minimal. Predictions about a core provide search guidance, not a refutation or certificate of unsatisfiability.

3.2.6 Architectural Capability Architectural capability covers algorithmic alignment *X6.1* and logic expressivity (*X6.2*). Algorithmic alignment concept refers to the degree to which a learning approach is compatible with PL syntax. In contrast, logic expressivity records the reported logical syntax or representation. It does not imply that one normal form has greater expressive power than another [Weitze and Hofacker \(1995\)](#).

Table 8 specifies the necessary evidence, sufficient coding evidence, exclusion criteria, and borderline-case decisions for assigning *X6* concepts to surveyed papers.

3.2.7 Soundness and Reliability Soundness guarantee (*X7*) records a guarantee, under stated assumptions, that every accepted NeSy reasoning or proving result in PL is logically valid with respect to the underlying PL semantics. The scope of the guarantee must be explicit, for example, a proof calculus. Neither predictive accuracy nor the mere availability of a certificate format establishes this guarantee.

Reliability (*X8*) includes robustness (*X8.1*) and interpretability (*X8.2*). Robustness refers to the ability of a NeSy framework to maintain reliable performance under variations, perturbations, or changes in the operating environment ([Balendran et al. 2025](#), p. 1). Interpretability denotes the extent to which a human can understand and reason about the internal structure and decision-making process of the NeSy framework [Pelosi et al. \(2025\)](#).

Table 9 specifies necessary and sufficient coding evidence, exclusion criteria, borderline-case decisions, and candidate examples for *X7* and *X8*, including *X8.1*, and *X8.2* concepts.

Table 8. Coding criteria for X6 concepts.

Code	Necessary evidence	Sufficient coding evidence	Exclusion criteria	Typical borderline case and decision	Candidate example
X 6	An architectural mechanism explicitly linked to the NeSy framework's PL computation.	Specified components and interactions establish how the architecture supports a PL operation or reasoning task.	Model-family names, parameter counts, or performance results without an architecture-to-computation link.	A standard architecture may qualify when its relevant computation and PL role are specified; naming it alone is insufficient.	×
X 6.1	An explicit operational correspondence between the neural architecture or learning procedure and PL syntax.	A documented mapping showing how PL variables, connectives, clauses, or formula structure determine neural components, connectivity, update rules, or learning operations.	Merely accepting PL inputs or naming a supported representation without an architecture or learning level correspondence.	Syntactic alignment qualifies when operationally demonstrated. Correspondence with a specified reasoning algorithm is not required. Representation support alone is insufficient for X 6.1 and should instead be considered under X 6.2.	Zhang et al. (2026)
X 6.2	A specified supported logical syntax or representation.	An operational account showing how that representation is encoded or processed, including restrictions.	Broad claims of logical expressiveness without a supported language.	CNF support qualifies as a representation characteristic, not evidence of greater expressive power than other PL forms.	Nagamatu et al. (2000)

Table 9. Coding criteria for X7 and X8 concepts.

Code	Necessary evidence	Sufficient coding evidence	Exclusion criteria	Typical borderline case and decision	Candidate example
X7	A universal, scoped soundness claim and stated assumptions.	A formal proof or explicit soundness-preservation argument covering the accepted outputs in scope.	Accuracy; finite testing; a component theorem with no justified connection to the accepted-output pathway.	A component guarantee supports only its stated scope, not unrestricted end-to-end soundness.	Seda (2006)
X8	An explicit reliability-related property with evidence supporting its stated scope.	A mechanism, analysis, or evaluation establishes the claimed reliability-related property within that scope.	Unqualified reliability claims; confidence scores without supporting analysis; merely restating an evidenced child property.	Interpretability evidence supporting X 8.2 does not also warrant X 8 unless a separate claim meets the direct-parent rule.	×
X8.1	A specified variation or perturbation and evidence of maintained performance.	An appropriate evaluation or formal result showing robustness under the stated conditions.	Standard accuracy without variation; untested claims of robustness.	Robustness to one perturbation type does not establish robustness to other shifts.	Alzaeemi and Sathasivam (2018)
X8.2	An explicit, intelligible correspondence between representations/decisions and human-understandable concepts.	A documented inspectable representation or interpretation method with supporting examples or evaluation.	Symbolic components, attention maps, or the word interpretable alone.	A faithful static representation may qualify; interaction is not required.	Riegel et al. (2020)

3.3 Y: NeSy Reasoning and Proving Assurance Level in PL

This section describes the Y dimension of the X–Y–Z taxonomy, which represents the assurance level (Y0) of a NeSy framework for reasoning and proving tasks in PL, as illustrated in Figure 4. Here, assurance level refers to the framework’s documented assurance characteristics, rather than an ordinal ranking. These characteristics include properties and mechanisms, such as soundness guarantees, proof validation, and certificate support, that provide grounds for confidence in the correctness and checkability of its

results. A framework is annotated with the corresponding *Y* concepts when the publication provides evidence of these characteristics. A framework need not exhibit every characteristic included in the *Y* dimension.

- Y0: Assurance level
 - Y1: Verification
 - Y1.1: Certificate support
 - Y1.1.1: Certificate format
 - Y1.1.2: External verification
 - Y1.2: Proof validation
 - Y1.3: Proof compression
 - Y1.4: Refutation certification
 - Y1.4.1: UNSAT certification
 - Y2: Interactive explanation
 - Y3: Soundness assessment

Figure 4. Y dimension: taxonomy for NeSy reasoning and proving assurance level

3.3.1 Verification In the taxonomy, verification (Y1) supports the assessment of correctness in NeSy reasoning and proving through mechanisms such as certificate support (Y1.1), proof validation (Y1.2), proof compression (Y1.3), and refutation certification (Y1.4).

Table 10 specifies necessary and sufficient coding evidence, exclusion criteria, and borderline-case decisions for Y1, Y1.1, Y1.1.1 and Y1.1.2 concepts. These criteria distinguish format support from actual checking. Supporting a certificate format does not establish that certificates were checked, and using a separate checker does not imply that the checker itself is formally verified.

Table 10. Coding criteria for Y1, Y1.1, Y1.1.1 and Y1.1.2 concepts.

Code	Necessary evidence	Sufficient coding evidence	Exclusion criteria	Typical borderline case and decision	Candidate example
Y1	A procedure checking PL results or formal evidence, or a documented mechanism facilitating such checking.	The checking procedure or enabling mechanism is operationally specified together with its target result or evidence.	Unexplained “verified” labels; evidence generation without checking support; solver answers alone.	A checking interface may establish facilitation without an executed check if its enabling role is operationally documented.	×
Y1.1	Support for an input-associated, machine-checkable evidence object.	The framework supports an evidence format with an input link and specified machine-checking semantics.	Result labels; unlinked traces; informal explanations or logs without machine-checkable semantics.	A supported checkable format need not have been checked in an experiment; an informal log without checking semantics is insufficient.	Ghanem (2024)
Y1.1.1	A concrete representation for machine-checkable evidence.	A named format or specified evidence structure associated with the framework’s outputs.	Generic statements that certificates could be supported.	Format support does not show that certificates were checked.	Ghanem (2024)
Y1.1.2	Checking by a verifier separate from the result-producing solver.	Reported execution or integration of that verifier against the input and evidence object.	Merely naming an available checker.	A separate checker need not be formally verified; record that distinction.	(Sun et al. 2026, Methods)

Certificate support (Y1.1) records support for a certificate, a machine-checkable evidence object associated with an input and a claimed result. For SAT, a model serves as a certificate and can be checked by evaluating the input formula. For UNSAT, a certificate encodes a refutation, often as a proof trace that is a structured record of proof-relevant steps in a specified checking format. A generic execution log, confidence score, or informal explanation does not by itself constitute a certificate. The narrower concepts distinguish certificate format (Y1.1.1) from external verification (Y1.1.2).

Certificate format (Y1.1.1) denotes the certificate representation used by a NeSy framework, for example DRAT format or the earlier DRUP format Wetzler et al. (2014). For formulas in conjunctive normal form (CNF), DRAT logs clause additions and deletions.

External verification (Y1.1.2) denotes checking a certificate against the original input formula using a verifier separate from the solver. For example, DRAT-trim checks DRAT refutations Wetzler et al. (2014). An independent checker is not necessarily a formally verified checker, as establishing its correctness is a separate assurance question Lammich (2020).

Proof validation (Y1.2) records actual checking of a proof or proof trace against the input and the rules of its proof system, including whether it establishes the claimed conclusion. Supporting a certificate format alone does not constitute validation.

Proof compression (Y1.3) refers to a method for efficient handling of formal proofs in PL Rollini et al. (2014); Gordeev and Haeusler (2019). Various compression techniques have been developed to reduce the size of an existing proof in PL, thereby improving the efficiency of proof verification. Foundational studies, such as Rollini et al. (2014), have specifically examined resolution-based proof compression in PL.

Proof validation (Y1.2) and proof compression (Y1.3) are sibling concepts under verification, because validation concerns checking correctness, whereas compression concerns reducing proof size while preserving validity.

Table 11 specifies necessary and sufficient coding evidence, exclusion criteria, borderline-case decisions, and candidate examples where available for proof validation (Y1.2) and proof compression (Y1.3).

Table 11. Coding criteria for Y1.2 and Y1.3 concepts.

Code	Necessary evidence	Sufficient coding evidence	Exclusion criteria	Typical borderline case and decision	Candidate example
Y1.2	Actual checking of proof steps and the claimed conclusion.	A reported validation procedure or validation results under a specified proof system.	Trace generation, format support, or informal inspection alone.	An internal proof checker qualifies here even if it does not qualify as external verification.	Amato et al. (2025)
Y1.3	A proof-size reduction transformation preserving validity and conclusion.	A documented validity-preserving compression method with a reported reduction or argument.	File compression; unchecked deletion of proof steps.	Compression does not imply that a checker was executed afterward.	×

Refutation certification (Y1.4) concerns checkable evidence for conclusions established by contradiction. Its child, UNSAT certification (Y1.4.1), records provision of a refutation certificate establishing that the input formula has no model, for example in DRAT format Wetzler et al. (2014). Certificate generation and successful certificate validation are distinct observations and are annotated accordingly.

Table 12 specifies the necessary and sufficient coding evidence, exclusion criteria, and borderline-case decision for Y1.4 and Y1.4.1, alongside a candidate example.

Table 12. Coding criteria for Y1.4 and Y1.4.1.

Code	Necessary evidence	Sufficient coding evidence	Exclusion criteria	Typical borderline case and decision	Candidate example
Y1.4	Checkable evidence supporting a conclusion established by contradiction.	An evidence object and specified checking relation establish how the contradiction-based conclusion can be validated.	Bare UNSAT labels; inconsistency predictions without supporting evidence; contradiction claims lacking documented checkability.	Refutation generation supports Y 1.4 only when the resulting evidence has a documented checking relation; an UNSAT response alone is insufficient.	×
Y1.4.1	Provision of a complete machine-checkable refutation certificate for an UNSAT claim.	A documented certificate-output mechanism in a specified checking format.	Bare UNSAT outputs; predicted cores; incomplete execution logs.	Certificate provision and successful certificate checking are separate observations.	Ghanem (2024)

3.3.2 Interactive Explanation and Soundness Assessment Interactive explanation (Y2) enables step by step, human interpretable traces for NeSy reasoning or proving result in PL. Unlike explainability approaches that focus on a PL formula of minimal size that produces the same truth value as the input PL formula Jaakkola et al. (2025), interactive explanation emphasizes explanations in natural language combined with symbolic traces Awolesi (2025). This concept is crucial for NeSy frameworks because it enhances the interpretability of the results of PL reasoning and proving for users and fosters trustworthy collaboration between humans and AI. Here, interactive means that users control which explanations are displayed, for example by expanding or collapsing steps or requesting clarification. Human-interpretable means that intended users can understand the reasoning steps and their connection to the result, which is also possible with non-interactive reports.

Soundness assessment (Y3) of a NeSy proof system refers to assessing whether every PL formula that the system can prove from a set of PL formulas is entailed by that set Smith (2020). If a NeSy proof system is sound, it cannot derive false conclusions from true premises. Whereas soundness guarantee (X7) records a reported guarantee of soundness, soundness assessment (Y3) records the documented methods and evidence used to assess or substantiate that claim. The same soundness proof may support both annotations, but an assessment alone does not establish a soundness guarantee.

Table 13 specifies evidence requirements, exclusions, and borderline decisions for coding interactive explanation (Y2) and soundness assessment (Y3). It distinguishes user-controlled explanations from static outputs and soundness assessment from predictive accuracy, clarifying that detecting unsoundness qualifies as assessment but does not establish a soundness guarantee.

Table 13. Coding criteria for Y2 and Y3 concepts.

Code	Necessary evidence	Sufficient coding evidence	Exclusion criteria	Typical borderline case and decision	Candidate example
Y2	User control over explanations linked to a particular result or reasoning process.	A documented interaction workflow, interface, or demonstration showing that connection.	Static explanations; unrelated input controls; generic chat interfaces.	An expandable explanatory proof viewer can qualify; a static proof PDF does not.	×
Y3	A documented investigation of truth preservation or semantic correctness within a stated scope.	A soundness proof, preservation analysis, counterexample search, or targeted semantic test.	Routine predictive accuracy without a soundness-specific assessment.	An assessment detecting unsoundness still documents assessment; record its negative outcome and do not assign X7.	Roth (1996)

3.4 Z: Evaluation Metric for NeSy Frameworks

This section describes the Z dimension which constitutes the evaluation metric (Z0), of the X–Y–Z taxonomy. Figure 5 illustrates its structure and constituent concepts. Concepts (Z1) and (Z2) represent numerical scores for reasoning and proving capability (X dimension) and assurance (Y dimension), respectively. Concept (Z3) represents joint coverage, which measures the combined breadth of capability and assurance concepts assigned to a paper.

- Z0: Evaluation metric
- Z1: Capability level
- Z2: Assurance level
- Z3: Joint coverage

Figure 5. Z dimension: evaluation metric for NeSy frameworks

In this survey, evaluation includes both qualitative analysis and quantitative assessment. Section 4 provides a qualitative comparison of existing literature reviews and their methodologies and taxonomies. Section 5 qualitatively analyses the surveyed NeSy frameworks and assigns X and Y concepts according to the capabilities and assurance properties or mechanisms reported in the corresponding publications. Section 6 subsequently provides a quantitative evaluation based on these annotations. Thus, the qualitative analysis in Section 5 supplies the data from which the Z scores are computed. The Z scores are therefore derived numerical measures, rather than concepts assigned directly during annotation. They are not used in Section 4

because that section compares literature reviews rather than evaluating individual NeSy frameworks using X and Y concepts. They are not reported in Section 5 because that section establishes and explains the annotations, whereas computing the scores requires corpus-wide concept frequencies and normalization values. The absence of Z scores from its annotation tables therefore reflects the separation between qualitative annotation and the subsequent quantitative evaluation in Section 6.

The capability, assurance, and joint coverage scores are computed using the formal definitions provided in Section 6.1. These scores provide corpus-relative summaries of the annotated capability and assurance profiles. The labels capability level and assurance level refer here to the derived numerical scores. They do not denote hierarchy depth or an ordering of taxonomy identifiers. A higher score does not by itself establish better task performance, stronger neurosymbolic integration, or greater soundness. In particular, joint coverage expresses how broadly a paper covers both dimensions relative to the maximum coverage observed in the surveyed corpus.

4 Existing Surveys

This section provides an overview of existing survey and review papers on NeSy AI. In particular, it examines the taxonomies proposed in these works and discusses how NeSy frameworks are classified using these taxonomies with respect to reasoning and proving capabilities and assurance levels in PL. The analysis is based on publications retrieved via a SPARQL query executed against the Digital Bibliography & Library Project (DBLP) SPARQL endpoint. The query and its results are publicly available in a GitLab repository, referenced in Listing 2 of Appendix A.1.

In total, 76 papers published between 1971 and 2025 were retrieved and analyzed. Of those, 26 are survey or review papers focused on NeSy frameworks. Tables 14 and 15 summarize the classification taxonomies used in 26 survey and review papers on NeSy AI and their alignment with the X-Y-Z taxonomy.

Figure 6 shows a Venn diagram that illustrates survey papers that use more than one classification system to categorize NeSy frameworks, based on data in Tables 14 and 15. The X and Y dimensions are also used to highlight the reasoning and proving capabilities or assurance levels in PL of the NeSy frameworks discussed in the research papers cited by these surveys and reviews. Notably, some survey and review articles listed in Tables 14, such as Colelough and Regli (2025), do not clearly explain the reasoning and proving capabilities or assurance levels in PL that are already discussed in the primary works they reference. The alignment between the classifications (taxonomies)

employed in these survey papers and the X and Y dimensions addresses this gap.

Table 14. Taxonomies employed in survey papers on NeSy AI and their alignment with the X–Y–Z taxonomy (continued in Table 15)

ID	Taxonomy name	Survey papers	Dimensions within taxonomy	X and Y concept
1.	PRISMA methodology Page et al. (2021) McGuinness (2021)	Colelough and Regli (2025)	1. Logic and reasoning	X1.1
			2. Learning and inference	X3.1.2.1
			3. Explainability & trustworthiness	Y2
2.	Task-directed NeSy taxonomy	Delvecchio et al. (2025)	4. Knowledge representation	Y3
			1. Rule mining	Y2
3.	H. Kautz’s taxonomy Kautz (2022)	Delong et al. (2025)	2. Rule enforcement	See Table 16
			3. Program synthesis	
4.	ESI nomenclature	Fitas (2025) Wan et al. (2024) Renkhoff et al. (2024) Hamilton et al. (2022) Bhuyan et al. (2024) Acharya et al. (2023) Lamb et al. (2020) (Hamilton et al. 2022, p. 25, Figure 28)	1. K-1:Symbolic Neuro Symbolic	X
			2. K-2:Symbolic[Neuro] 3. K-3:Neuro[Symbolic] 4. K-4:Neuro:Symbolic → Neuro 5. K-5:Neuro_[Symbolic] 6. K-6:Neuro[Symbolic]	
5.	LLR taxonomy	Delong et al. (2025)	1. Ensemble 2. Sequential 3. Integrated 3.1 Nested 3.2 Cooperative 3.3 Compiled	See Table 16
			1. Logically informed embedding approaches 2. Learning With Logical Constraints 3. Rule learning for knowledge graph completion	
6.	KR taxonomy	Cheng et al. (2025)	1. Knowledge graph completion	X1.1
7.	Natural language processing taxonomy	Aker et al. (2025)	2. Rule learning dimension	X3.2, X3.1.1
			1. Tasks	X1.1,
8.	SWOT methodology	Nawaz et al. (2025)	2. Paradigm	X5.1, X3.1.1
			1. Strengths	X1.1
9.	G-L-based representation	Bizzarri et al. (2025)	2. Weaknesses	X2.1, X3.1
			3. Opportunities	
10.	A3PO taxonomy	Li et al. (2024)	4. Threats	Y3
			1. Graph-based representation	
11.	NeSy four-dimensional taxonomy	Wang et al. (2024b)	2. Logic-based representation	X
			1. Autoformalization 2. Premise selection 3. Proof-step generation 4. Proof search 5. Other tasks	
			1. Functionality	X
			2. Knowledge embedding 3. Knowledge representation 4. Neural-symbolic integration	

The Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA) 2020 framework is a widely adopted methodology for reporting systematic reviews Page et al. (2021). According to the most recent iteration of the PRISMA 2020 statement, the checklist comprises 27 items, while the abstract section is subject to 12 additional requirements McGuinness

Table 15. Taxonomies employed in survey papers on NeSy AI. Alignment taxonomies used in the survey papers with X–Y–Z taxonomy.

ID	Taxonomy name	Survey papers	Dimensions within taxonomy	X and Y concept
12.	NeSy-7 taxonomy	Marra et al. (2024)	1. The approach to logical inference. 2. The syntax of the logical theories used. 3. The logical semantics of the systems and their extensions to facilitate learning. 4. The scope of learning, encompassing either parameter or structure learning. 5. The presence of symbolic and subsymbolic representations. 6. The degree to which systems capture the original logic, probabilistic, and neural paradigms. 7. The classes of learning tasks that the systems perform	×
13.	Yu's taxonomy	Yu et al. (2023) Renkhoff et al. (2024) Hamilton et al. (2022) Acharya et al. (2023)	1. Learning for reasoning 2. Reasoning for learning 3. Learning-reasoning	See Table 17 See Table 18
14.	KLREA taxonomy	Gibaut et al. (2023)	1. Knowledge representation 2. Learning 3. Reasoning 4. Explainability and trustworthiness 5. Applications	×
15.	×	Besold et al. (2021)	×	×
16.	×	Berlot-Attwell (2021)	×	×
17.	KRERL taxonomy	(Garcez et al. 2019, p. 627, Figure 5)	1. Knowledge Representation 2. Extraction 3. Reasoning 4. Learning	×
18.	ADT taxonomy	Townsend et al. (2019) Andrews et al. (1995)	1. Quality 2. Expressive power 3. Translucency of the network 4. Portability 5. Complexity	×
19.	ILU taxonomy	Bader and Hitzler (2005) Acharya et al. (2023) Bouneffouf and Aggarwal (2022)	1. Interrelation 2. Language 3. Usage	×

(2021). The objective of these modifications is to ensure the transparency of scholarly review methodologies. Using the PRISMA methodology, the authors of [Colelough and Regli \(2025\)](#) conducted a systematic review of NeSy AI projects published between 2020 and 2024. The review is structured along four fundamental dimensions, as outlined in Table 14. The authors placed particular emphasis on logic and reasoning within the broader context of NeSy AI. In the context of the aforementioned dimension, the authors undertake a comprehensive examination of eight NeSy frameworks ([Colelough and Regli 2025, p. 6, Figure 2](#)). The following trends in NeSy AI research were identified as being of particular significance for logic and reasoning. First, the integration of formal logic and probabilistic models is essential for supporting more robust decision-making processes (X1.1: [satisfiability decision](#); Y3: [soundness assessment](#)). Second, the incorporation of metacognitive mechanisms should help NeSy frameworks adapt their reasoning strategies (X3.1.2.1: [strategy monitoring](#)). This adaptation should enhance the interpretability and clarity of explanations (Y2: [interactive explanation](#)).

However, the review does not offer a detailed assessment of which PL reasoning or proving methods (X3.1.1: [solution method](#)) or which types of formal verification (Y1: [verification](#)) are actually supported by the reviewed NeSy frameworks. For example, the authors cite [Chanin and Hunter \(2023\)](#) to illustrate the integration of NeSy frameworks with autonomous systems, but in the same work [Chanin and Hunter \(2023\)](#), to support decision-making, the authors implemented a resolution procedure with non-binary unification (X3.1.1.2: [resolution](#)) and used a NeSy theorem prover to query formulas expressed in a formal logic and to conduct reasoning (X5.1: [logical generalization](#); X3.1.1: [solution method](#)) over these formulas derived from text written in a natural language.

The task-directed NeSy taxonomy proposed in [Delvecchio et al. \(2025\)](#) classifies NeSy approaches into three categories: rule mining, rule enforcement, and program synthesis. This taxonomy provides the organizing structure for surveying papers published between 2017 and 2024. A central research question in the survey concerns how symbolic components can be integrated with neural networks such as solvers to enhance explainability (Y2: [interactive explanation](#)) and reasoning capabilities. The survey does not focus exclusively on NeSy reasoning or theorem proving in PL. In the context of rule mining, the survey highlights the role of Horn clauses, originally developed in PL, which serve to represent structured knowledge and enable reasoning across domains such as medicine. The survey also discusses propositionalization, understood as the transformation of relational data into vectorized representations suitable for neural learning. Crucially, propositionalization is not equivalent to translating relational data into PL formulas. Within the rule enforcement category, the survey discusses NeSy regularization strategies [Ahmed et al. \(2022\)](#), as well as reasoning approaches based on first-order logic ([Barwise 1977](#), p. 17, Section 3) and probabilistic logic [Nilsson \(1986\)](#), particularly in the context of enhancing explainability. Finally, within the program synthesis category, the survey describes examples of using the Prover9 solver [McCune \(2005–2010\)](#) in semantic parsing pipelines.

According to Kautz’s taxonomy, NeSy architectures are classified into six types [Kautz \(2022\)](#). For an overview of the six types of NeSy architectures, see the summary in the first paragraph of Section 3.3 of [Bizzarri et al. \(2025\)](#). For a more detailed exposition of this topic, please refer to the original work [Kautz \(2022\)](#). As illustrated in Table 14, eight scholarly articles are enumerated that adapt Kautz’s taxonomy (from K-1 to K-6) to survey NeSy frameworks. An exhaustive exposition of the mappings between Kautz’s taxonomy and the X and Y concepts is provided in Table 16, encompassing seven of these eight survey papers.

Table 16. Mapping between categories of Kautz’s taxonomy and the X and Y concepts across survey papers that employ Kautz’s taxonomy.

ID	Reference	K-1	K-2	K-3	K-4	K-5	K-6	Comment
1	Delong et al. (2025)	×	×	×	X1.1, X3.1.1	×	X3.1.1	
2	Fitas (2025)	×	×	×	×	×	×	
3	Wan et al. (2024)	×	×	×	×	×	X8.1, X8.2,	
4	Renkhoff et al. (2024)	×	×	X3.1.1, X1.1 Y3, X7	×	×	X1.1, X3.1.1	
5	Hamilton et al. (2022)	×	×	×	×	×	×	
6	Bhuyan et al. (2024)	X6.2	×	×	X1.1	×	×	see Table 20
7	Lamb et al. (2020)	×	×	×	×	×	×	

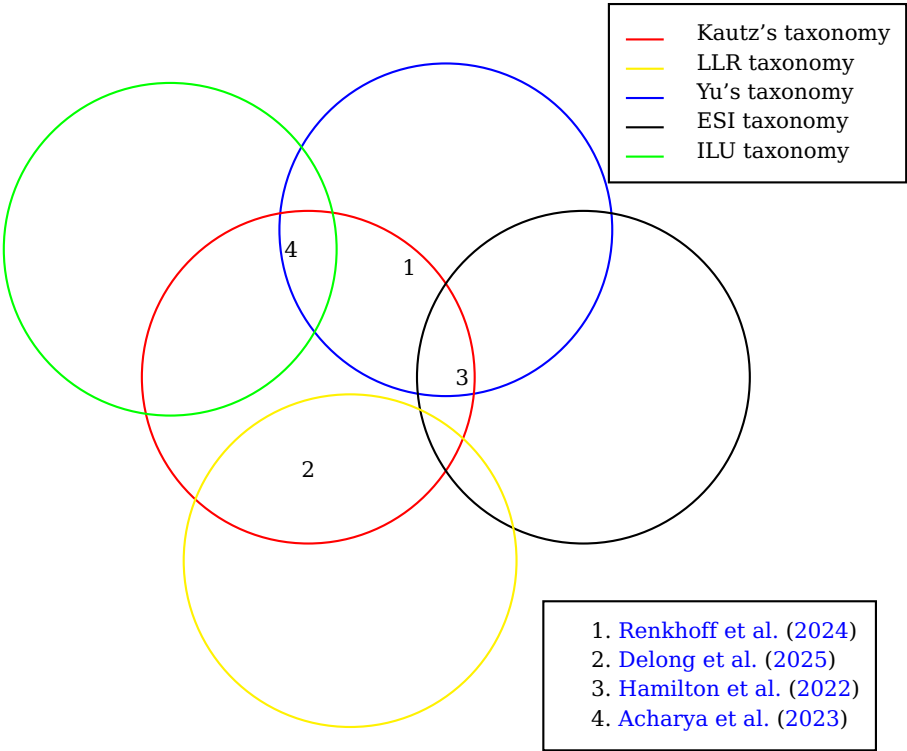


Figure 6. Venn diagram illustrating survey papers that use multiple taxonomies to categorize NeSy frameworks, derived from [Tables 14 and 15](#).

A comprehensive survey into the application of NeSy AI methodologies for reasoning over knowledge graphs is presented in the study [Delong et al. \(2025\)](#). The authors propose a novel taxonomy to describe NeSy AI

approaches for reasoning over knowledge graphs. As illustrated in Figure 1 of the survey [Delong et al. \(2025\)](#) and listed in entry 5 of Table 14, this taxonomy is organized into three main categories: logically informed embedding approaches, learning with logical constraints, and rule learning for knowledge graph completion.

Each category is comprised of one or more subgroups, which correspond to specific types of Kautz’s taxonomy. To reference these subgroups in this paper, the notation `category_id(K- $n$ )` is used, where n denotes the associated type in Kautz’s taxonomy. For instance, `A(K-3)` denotes a group of NeSy frameworks within Kautz’s type K-3 category that corresponds to category A. In the background section, the survey describes PL Horn clauses. Within the `A(K-4)` subgroup, the authors highlight how Horn clauses and the forward-chaining method (X3.1.1: solution method) are used in knowledge graph augmentation ([Delong et al. 2025](#), Subsection IV-A.2). The authors discuss the application of Horn clauses in rule mining ([Delong et al. 2025](#), Subsection IV-C). There is no evidence in the survey paper [Delong et al. \(2025\)](#) that Horn clauses expressed in PL play a role in rule mining or knowledge graph augmentation. Within the `B(K-6)` subgroup, the authors cite work on the graph collaborative reasoning method [Chen et al. \(2022\)](#), which translates knowledge graph structure into formulas in a formal logic ([Delong et al. 2025](#), Subsection IV-B.2). In the survey [Delong et al. \(2025\)](#) the authors do not specify that these formulas translated from knowledge graphs are expressed in PL. Theorem 1 in ([Chen et al. 2022](#), p. 77, Theorem 1) establishes the conditions under which these PL formulas are true or false (X3.1.1: solution method). However, the survey [Delong et al. \(2025\)](#) does not examine how theorem proving methods can be applied to knowledge graphs in a NeSy theorem proving context. In the broader NeSy sense, the application of theorem proving techniques in knowledge graphs still remains largely underexplored, particularly in the context of verifying the soundness (Y3 : soundness assessment). The utilization of such methodologies has the potential to facilitate tasks such as rule-guided embeddings ([Delong et al. 2025](#), Subsection IV-A.2) by facilitating the implementation of methods for solving Horn constraints [Unno et al. \(2017\)](#). Furthermore, the integration of proof compression techniques (Y1.3: proof compression) can directly reduce the size of derived proofs in PL. As demonstrated in [Heule et al. \(2013\)](#), causal-proof trimming and related optimizations yield shorter proofs in PL ([Heule et al. 2013](#), Section III). These optimizations have the potential to enhance the interpretability and verifiability of reasoning processes applied to knowledge graphs. In addition, an important study on theorem proving over knowledge graphs [Li et al. \(2025\)](#), introduces KG-Prover, which integrates a mathematical knowledge

graph with a large language model and the Lean proof assistant [de Moura et al. \(2015\)](#) to generate and verify formal proofs (Y1.2: proof validation).

In the paper [Fitas \(2025\)](#), the author undertakes an extensive review of NeSy AI techniques and their applications in signal and image processing. The survey is organized according to Kautz’s six dimensions, which are based on the level of integration between symbolic reasoning and neural networks (see [Table 14](#), and [Table 16](#)). In the review of NeSy AI applications in multimedia, vision, and entertainment, the author describes a NeSy framework, called Laser [Huang et al. \(2025\)](#). This framework aligns raw video clips with PL for the purpose of improving semantic video understanding. However, in this specific instance, the author does not expound upon the manner in which NeSy reasoning in PL is manifested within the Laser framework. The present review makes no mention of NeSy theorem proving in PL.

The paper [Wan et al. \(2024\)](#) provides a concise survey of cognitive AI systems, focusing on NeSy AI algorithms. While Kautz’s taxonomy [Kautz \(2022\)](#) defines six NeSy architectures, the survey employs a subset of five categories to structure its analysis of NeSy AI algorithms. This classification serves as a useful entry point for readers seeking to navigate the landscape of NeSy algorithms. However, the survey does not undertake a comprehensive examination of NeSy reasoning capabilities. Additionally, it does not address NeSy theorem proving as a discrete research domain. For instance, within the K-5 category, the authors cite [Garcez et al. \(2019\)](#), but they do not discuss the propositionalization component emphasized in that paper. The importance of explainability (X8.2: interpretability) and robustness (X8.1: robustness) of symbolic reasoning is discussed within the K-6 category. *The survey is relevant to reasoning-focused NeSy research, but it provides limited detail about how the reviewed systems perform reasoning.*

The paper [Lamb et al. \(2020\)](#) adopts Kautz’s taxonomy to review the use of GNNs as a model for NeSy computing, including the review of the application of GNNs in different domains. However, inference, reasoning, and proving in PL are not explicitly discussed within the categories of Kautz’s taxonomy (see [Table 16](#)), but they are discussed in the context of the relationship between GNNs and NeSy computing ([Lamb et al. 2020](#), p. 2-5, Section 3).

However, [Table 15](#) lists four papers that utilize Yu’s taxonomy [Yu et al. \(2023\)](#). [Table 17](#), in turn, illustrates methods that utilize PL in NeSy frameworks described in [Yu et al. \(2023\)](#). [Table 18](#) presents a detailed mapping between Yu’s taxonomy and the X and Y concepts across survey papers that adopt Yu’s taxonomy. Yu’s taxonomy is predicated on the integration mode between neural and symbolic systems. The tree in [Figure 7](#) presents Yu’s taxonomy, which classifies NeSy learning frameworks into

three distinct categories: learning for reasoning, reasoning for learning, and learning–reasoning (Yu et al. 2023, p. 109, Table 2). Each category is associated with one or more methods, represented as leaf nodes in the tree shown in Figure 7.

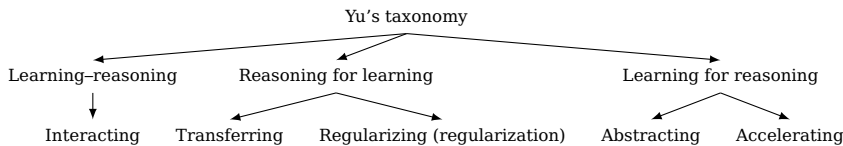


Figure 7. Methods assigned to categories in Yu’s taxonomy, derived from (Yu et al. 2023, p. 109, Table 2)

Table 17. Methods that utilize PL in NeSy frameworks within categories across Yu’s taxonomy Yu et al. (2023).

ID	Reference	Learning for Reasoning	Reasoning for Learning	Learning- Reasoning
1	Xu et al. (2018)	×	regularization (semantic loss)	×
2	Silva and Gombolay (2021)	×	transferring	×

In survey (Yu et al. 2023, pp. 113-114, Section 3.2.1.), the authors describe semantic loss as a regularization approach within the reasoning for learning category (see Figure 7), stating that it utilizes PL reasoning (X1.2: logical consequence, see Table 18) to improve the learning ability of neural networks Xu et al. (2018). However, the survey does not mention which PL reasoning methods are employed. A closer look at the original paper that describes semantic loss reveals that the PL statement, expressed in CNF, is used as input to weighted model counting (WMC) procedure (Belle 2017, p. 4, Section 2.2) to compute semantic loss (Xu et al. 2018, p. 114, Section 5.1, Figure 3). From a formal logic perspective, WMC is a PL inference method rather than a reasoning method Chavira and Darwiche (2008).

The survey Acharya et al. (2023) uses Yu’s taxonomy to categorize NeSy reinforcement learning frameworks. However, as illustrated in Table 14, the survey also highlights Kautz’s taxonomy (Acharya et al. 2023, p. 1942, see Table I, Table II). The survey describes work on Propositional Logic Nets (PROLONETS) Silva and Gombolay (2021) that encodes domain knowledge written in PL into a neural network, but the authors do not describe the reasoning and proving capability or assurance level of such NeSy reinforcement learning frameworks, which is marked as × in Table 18.

However, in the survey (Yu et al. 2023, p. 116-117, Section 3.2.1., Figure 13) the authors describe NeSy framework Silva and Gombolay (2021) that utilizes a transferring method within the reasoning for learning category of Yu’s taxonomy (see Table 17), and the authors claim that PROLONETS encodes PL statements in neural networks to improve their reasoning capabilities.

Table 18. Mapping between Yu’s taxonomy and the X and Y concepts across survey papers that employ Yu’s taxonomy Yu et al. (2023).

ID	Reference	Learning for Reasoning	Reasoning for Learning	Learning- Reasoning
1	Yu et al. (2023)	×	X1.2	×
2	Acharya et al. (2023)	×	×	×
3	Renkhoff et al. (2024)	X1.1, X3.1.1	×	X1.1

The survey Renkhoff et al. (2024) aligns the taxonomies proposed by Kautz (2022) and Yu et al. (2023) to describe how NeSy frameworks support validation & verification, as well as testing & evaluation. Table 19 provides a comprehensive overview of the aforementioned alignment, presenting the NeSy frameworks belonging to categories of Kautz’s and Yu’s taxonomies, as categorized in the survey Renkhoff et al. (2024). For instance, the original work Manhaeve et al. (2018) introduces DeepProbLog as a programming language that integrates probabilistic logic inference and deep learning techniques.

Table 19. NeSy frameworks classified according to the taxonomies of Kautz (2022) and Yu et al. (2023), as reported in the survey Renkhoff et al. (2024).

ID	Kautz’s taxonomy	Yu’s taxonomy	NeSy framework
1.	K-2:Symbolic[Neuro]	Learning for Reasoning	AlphaGo Zero Silver et al. (2017)
2.	K-3:Neuro Symbolic	Learning for Reasoning or Learning-Reasoning	DeepProbLog Manhaeve et al. (2018)
3.	K-6:Neuro[Symbolic]	Learning for Reasoning or Learning-Reasoning	Shield Alshiekh et al. (2018)

However, the survey (Yu et al. 2023, p. 5, Table 2) classifies DeepProbLog under the Learning-Reasoning category, whereas (Renkhoff et al. 2024, p. 10) places it in the Learning for Reasoning category. This flexibility in categorizing NeSy frameworks underscores the interpretative nature of such assignments, which often supersedes absolute interpretation. Readers,

including PhD students and practitioners, should treat these classifications as informed estimations or approximations, reflecting the reasoning behind why a particular NeSy framework fits into a certain category. Since different authors may apply different criteria, mismatches in taxonomy assignments are to be expected.

The survey (Renkhoff et al. 2024, p. 2) characterizes validation and verification as processes that ensure software meets required quality and reliability standards. Testing and evaluation are described as methods for conducting these processes (Y1.1: certificate support). Within this context, PL is introduced as a logical foundation that supports validation & verification (Y1.2: proof validation; Y1.1: certificate support). To establish this connection with precision, the authors map validation and verification tasks to corresponding reasoning mechanisms in PL. The validation of the symbolic component of NeSy AI is tied to logical concepts in PL such as validity and soundness (Y3: soundness assessment; X7: soundness guarantee). Other PL-related concepts, including decidability, truth tables, and semantic tableaux, are also discussed in (Renkhoff et al. 2024, Section IV-A). Notably, semantic tableaux in PL can be mapped both to fundamental decision procedures in PL (X1.1: satisfiability decision) and to reasoning methods (X3.1.1: solution method) concepts. These concepts are also used in the verification of the symbolic part of NeSy AI (Renkhoff et al. 2024, Sections IV-B). In summary, the survey is pertinent to the broader AI community because it illuminates the manner in which reasoning mechanisms in PL support validation and verification of NeSy frameworks. As an illustrative example, the authors cite DeepProbLog (Renkhoff et al. 2024, p. 10) to show how semantic tableaux in PL (X3.1.1: solution method) can be applied within the verification process. While the survey clearly demonstrates the role of PL-based reasoning in NeSy frameworks such as DeepProbLog, this connection becomes less explicit when considering other NeSy approaches discussed in the survey. Following the categorization reported in the survey Renkhoff et al. (2024), the work Alshiekh et al. (2018) is classified under the K-6 and Learning for Reasoning (Learning-Reasoning) categories (see Table 19). Based on the classification of NeSy frameworks in Renkhoff et al. (2024), mappings from the corresponding K-6 and Learning for Reasoning (Learning-Reasoning) categories to the X and Y concepts are illustrated in Table 16 and Table 18. In the aforementioned work Alshiekh et al. (2018), a reactive system, known as Shield, is utilized as a computational model that monitors the learning process and performs corrective actions when a chosen action violates a predetermined learning specification (Alshiekh et al. 2018, see Abstract). Notably, the survey does not address the fact that propositions in PL are used to model the inputs and the outputs of the reactive system (Alshiekh et al. 2018, p. 3). The truth value

assignment (X1.1: satisfiability decision) is employed to validate the inputs and the outputs of the reactive system.

The structured review [Hamilton et al. \(2022\)](#) examines NeSy frameworks for natural language processing (NLP). The review aims to answer the question of whether these NeSy frameworks meet the authors' descriptions and promises, which include reasoning, out-of-distribution generalization, interpretability, effective learning and reasoning from limited data, and transferability across domains ([Hamilton et al. 2022](#), see Abstract). The review incorporates Kautz's taxonomy within a nomenclature (see Table 14) to categorize NeSy frameworks. However, the review highlights Yu's taxonomy. Readers are referred to Figures 28–34 in ([Hamilton et al. 2022](#), pp. 25–28), which provide clear and unambiguous visualizations of all categories within the nomenclature used for categorizing NeSy frameworks for NLP. In the review, the authors clearly divide NeSy reasoning into model-based and theorem proving approaches ([Hamilton et al. 2022](#), p. 8). However, authors reveal in the review that there are discrepancies in the definition of reasoning and variation in its implementation within NeSy AI. The authors cite early NeSy frameworks such as KBANN [Towell and Shavlik \(1994\)](#) and CILP d'Avila Garcez et al. (2012) that embed PL in neural networks ([Hamilton et al. 2022](#), Figures 28–34, p. 7). However, authors cited SHRUTI [Wendelken and Shastri \(2004\)](#) and Neural Theorem Prover (NTP) [Rocktäschel and Riedel \(2016\)](#) as NeSy theorem-proving frameworks, but authors did not describe what type of proving these frameworks can perform.

The survey [Bhuyan et al. \(2024\)](#) provides an overview of NeSy implementations, categorizing them by integration type, application domain, system properties, and contribution category, as illustrated in ([Bhuyan et al. 2024](#), p. 12813, Figure 3). Reasoning paradigms such as deductive, inductive, combinatorial, common sense, and abductive [Garcez et al. \(2007\)](#) are described under the system properties category ([Bhuyan et al. 2024](#), p. 12817, Figure 6). To organize NeSy implementations by integration type, the authors adopt Kautz's taxonomy [Kautz \(2022\)](#). For each category in this taxonomy, the survey describes how individual NeSy frameworks employ these reasoning methods. Based on the categorization given in [Bhuyan et al. \(2024\)](#), Table 20 summarizes the reasoning paradigms and the number of cited works associated with PL across Kautz's taxonomy. An unspecified reasoning paradigm is marked with ×. Moreover, the table illustrates the mappings from PL-based reasoning methods utilized by these NeSy frameworks to X and Y concepts. Overall, Table 20 shows that the survey [Bhuyan et al. \(2024\)](#) cites 28 papers that are associated with PL, either as a representational language or as an active component of the NeSy reasoning process.

Table 20. Reasoning paradigms or methods utilized in NeSy frameworks across Kautz’s taxonomy based on survey [Bhuyan et al. \(2024\)](#) .

ID	Category in Kautz’s taxonomy	Inference / reasoning paradigm / method	Number of NeSy frameworks cited in Bhuyan et al. (2024) that are associated with PL	NeSy frameworks cited in Bhuyan et al. (2024) that use PL reasoning	X and Y concept
1.	K-1	Deductive	0	Staffa et al. (2011)	X6.2
		Common sense	0	×	×
2.	K-2	Case-based	6	×	×
		×	2	×	×
3.	K-3	Deductive	2	×	×
		×	1	×	×
4.	K-4	Constraint-based	0	×	×
		Case-based	2	×	×
		Neurule	1	×	×
		×	9	Hatzilygeroudis and Prentzas (2000)	X1.1
5.	K-5	Case-based	2	×	×
		Deductive	1	×	×
		Common Sense	1	×	×
		×	1	×	×
6.	K-6	×	0	×	×
TOTAL:			28		

The authors highlight that K-1 type NeSy frameworks employ deductive ([Bhuyan et al. 2024](#), p. 12819, Section 2.3.1) and common sense ([Bhuyan et al. 2024](#), p. 12826, Table 7) reasoning methods (see ID 1 in Table 20). However, the role of PL in NeSy reasoning is not explicitly discussed within the K-1 category of Kautz’s taxonomy, as indicated by the zero entries in the fourth column of Table 20. The survey also does not cite any NeSy frameworks that leverage a formal logic for deductive reasoning or in connection to common sense reasoning ([Bhuyan et al. 2024](#), p. 12826, row 4 in Table 7). Notably, the survey cites [Staffa et al. \(2011\)](#) as a K-1 type of NeSy framework. However, it does not mention the use of PL reasoning or representation in that framework. In contrast, [Staffa et al. \(2011\)](#) clearly describes using PL as a basis for reasoning in this framework by translating PL into a neural network implementation (X6.2: logic expressivity) ([Staffa et al. 2011](#), p. 2). Therefore, the fifth column in Table 20 contains the reference to [Staffa et al. \(2011\)](#) to reflect its use of PL. The authors cite eight NeSy frameworks within the K-2 category and claim that they use PL ([Bhuyan et al. 2024](#), p. 12827, Table 8). Of the eight cited NeSy frameworks, two do not have associated reasoning methods (marked as × in Table 20 and in ([Bhuyan et al. 2024](#), p. 12827, Table 8)). As illustrated in Table 20, three NeSy frameworks are cited in the survey ([Bhuyan et al. 2024](#), p. 12828, Table 9) and are thus classified within the K-3 category of Kautz’s taxonomy. Of the three NeSy frameworks under consideration, two have been assigned to PL and employ a deductive reasoning paradigm. The survey cites [Kraetzschmar et al. \(1998\)](#) and asserts that the framework employs PL. However, the underlying reasoning paradigm remains unspecified ([Bhuyan et al. 2024](#), Reasoning column, row 1 marked as × in Table 9). Notably, the NeSy frameworks associated with PL in the survey ([Bhuyan et al. 2024](#), p. 12827, Table 8; p. 12828, Table 9) do not actually

employ PL reasoning methods. These examples demonstrate that readers should not equate the presence of symbolic components with a reasoning method in a NeSy framework. The K-4 category encompasses a total of 12 cited papers in the domain of NeSy AI, all of which are associated with PL (Bhuyan et al. 2024, p. 12829, Table 10). As shown in Table 20, nine of the twelve cited NeSy frameworks do not have an associated reasoning method. However, within the K-4 category, the authors cite the work in Hatzilygeroudis and Prentzas (2000) and classify it as a NeSy framework without reasoning support. This assignment is questionable, as that work employs DNF in PL to express rules and truth tables (X1.1: satisfiability decision), which function as a reasoning mechanism in the training set specification process (Hatzilygeroudis and Prentzas 2000, p. 7, Section 3.2). In survey (Bhuyan et al. 2024, p. 12829, Table 10), Neurule is categorized as an inference method (see Table 20). However, Neurule is a framework that employs a hybrid inference mechanism based on a backward-chaining strategy (Hatzilygeroudis and Prentzas 2000, p. 11, Section 4.1, Section 4.3). The survey cites five NeSy frameworks belonging to the K-5 category that are associated with PL (Bhuyan et al. 2024, p. 12830, Table 11). As illustrated in Table 20, four of these aforementioned NeSy frameworks have been assigned a reasoning paradigm or method. The remaining framework has not been assigned a reasoning paradigm and is marked with $\times$. Notably, not every framework in the K-6 category employs PL or reasoning methods in PL. In the context of Kautz’s taxonomy, the existing literature includes studies on the encoding of PL within the KBANN neural architecture Towell and Shavlik (1994) and on embedding PL utilizing the translation technique CIL^2P Garcez and Zaverucha (1999). However, these two NeSy frameworks are not included in Kautz’s taxonomy (Bhuyan et al. 2024, p. 12820, Table 6). The survey paper makes no mention of theorem proving as a component of any NeSy framework.

Nevertheless, a thorough examination of NeSy knowledge graph reasoning methods is provided in Cheng et al. (2025), thereby complementing the previously discussed survey DeLong et al. (2025) (see ID 1 in Table 16). The authors of the study propose a taxonomy that is organized into two broad categories: knowledge graph completion and rule learning (Cheng et al. 2025, p. 225:4, Figure 2). The work systematically summarizes the intersection of neural networks and symbolic reasoning for knowledge graph reasoning tasks (Cheng et al. 2025, p. 225:3, Table 1). The survey also introduces PL (Boolean) reasoning (X1.1: satisfiability decision) as part of the symbolic reasoning section. It is acknowledged that PL possesses the theoretical capacity to support knowledge graph completion. The authors further highlight the role of the propositional maximum satisfiability problem (MaxSAT) (Bonet et al.

2007, see Abstract) (X3.2: MaxSAT optimization) method as an optimization-oriented reasoning technique. The survey's findings within the rule learning category indicate that there is no evidence of the application of NeSy theorem proving techniques in the joint learning of embeddings of symbols and interpretable logical rules from knowledge graphs (Cheng et al. 2025, p. 32).

A review of NeSy approaches in the domain of natural language processing (NLP) is presented by Akter et al. (2025). The authors have structured their review into tasks and paradigm dimensions, as illustrated in Figure 1 of their work Akter et al. (2025). Within the paradigm dimension, the review highlights methodologies that integrate a formal logic into neural networks to enhance NeSy reasoning capabilities (X5.1: logical generalization). For example, some approaches incorporate the PL implication $A \rightarrow B$ with the goal of enabling neural networks to approximate truth value assignment (X1.1: satisfiability decision). In another line of work, discussed by the authors, a proposition $A \wedge B$ is represented as a vector, and a neural network is then trained to learn the logical conjunction operation as a function, denoted as $f(A, B)$. This function effectively implements PL reasoning within a vector space (X3.1.1: solution method). By combining the reasoning power of symbolic logic with the flexibility and learning capacity of neural networks, these methods enable NeSy PL reasoning (X5.1: logical generalization) in NLP tasks.

As was the case in the preceding review, the present one addresses the impact of NeSy frameworks on advanced cognitive systems in domains such as NLP, robotics, and decision-making Nawaz et al. (2025). The review employs the SWOT (strengths, weaknesses, opportunities, threats) methodology to analyze five extant NeSy AI models. In the context of NeSy reasoning and theorem proving, the present review focuses on NeSy reasoning based on first-order and probabilistic logics, as well as neural theorem proving. Nonetheless, the authors cited work that employs the DeepProbLog model Manhaeve et al. (2018). However, they do not delve into the role of PL reasoning within DeepProbLog. In the aforementioned work, the ground logic program is rewritten in PL to define the truth value (X1.1: satisfiability decision) of the query in terms of the truth values of probabilistic facts (Manhaeve et al. 2018, p. 7, Section 4.1). In the review, the authors also emphasize the innovative nature of neural theorem proving methods, which integrate neural networks with theorem proving, contingent upon formal logic. However, the review does not emphasize the central role of PL reasoning in neural theorem proving approaches (X1.1: satisfiability decision). The authors cited the work of Riegel et al. (2020) as being directly relevant to neural theorem proving. The research demonstrates how NeSy reasoning grounded in PL is embedded into the model architecture of

logical neural networks (X2.1: satisfiability witness generation; X3.1: logical strategy). This integration enables the convergence of an inference algorithm with recursive directional graph traversal (Algorithm 3 in [Riegel et al. \(2020\)](#)) in finite time for the PL case (Y3: soundness assessment).

A survey of NeSy approaches for Network Intrusion Detection Systems (NIDS), accompanied by a critical analysis, is presented in [Bizzarri et al. \(2025\)](#). The authors primarily employ a two-dimensional categorization scheme to differentiate NeSy frameworks, distinguishing between logic-based and graph-based representations. In the survey, the authors describe the usage of PL in the NIDS frameworks to express rules in if-then form, that is, as PL implications ($p \rightarrow q$). For instance, the authors cited a NeSy framework named ODXU [Sander et al. \(2024\)](#) that utilizes PL to express decision trees, thereby enhancing the decision-making process. Nevertheless, the utilization of NeSy reasoning and proving capability or assurance level in PL within NIDS remains unaddressed.

The paper [Li et al. \(2024\)](#) presents a survey of deep learning for theorem proving. The authors highlight that Boolean satisfiability (SAT) solvers play a key role in theorem proving by handling PL, but in the rest of the paper, they do not discuss the proving assurance mechanisms in PL supported by NeSy frameworks.

A four-dimensional taxonomy is employed in the survey by [Wang et al. \(2024b\)](#) to classify NeSy frameworks. One of these dimensions, knowledge representation, includes the use of PL. Notably, the survey does not specify how NeSy frameworks realize reasoning capabilities or achieve particular proving assurance levels. Instead, PL is discussed primarily in terms of its contribution to improving the performance of semi-supervised classification and its capacity to express rules.

The paper [Marra et al. \(2024\)](#) surveys how reasoning and neural networks are combined, as well as how formal logic is combined with graphical probabilistic models (called StatAI frameworks). Authors use seven shared dimensions (see ID 12 in Table 15) to characterize these frameworks. The authors describe the SAT problem for PL ([Marra et al. 2024](#), p. 4, Section 2) as one of the most fundamental problems in computer science, but they did not explore how the reasoning and proving capability or assurance level in PL affects the properties of NeSy or StatAI frameworks.

The paper [Gibaut et al. \(2023\)](#) surveys NeSy frameworks from five different angles (see ID 14 in Table 15). The authors explain that Logical Boltzmann Machines (LBM) [Tran and Garcez \(2023\)](#) can represent any PL formula ([Gibaut et al. 2023](#), p. 5, Section 2). However, this topic is discussed in detail in Section 5 of this paper. Overall, the authors do not discuss how

well reasoning and proving capability or assurance level in PL is achieved in surveyed NeSy frameworks.

A comprehensive survey on NeSy learning and reasoning is given in [Besold et al. \(2021\)](#). PL is discussed as a means of expressing problems in neural networks as sets of weighted PL formulas ([Besold et al. 2021](#), p. 23, Section 5.3.1). However, NeSy reasoning and proving capabilities or assurance levels in PL are not discussed.

The paper [Berlot-Attwell \(2021\)](#) surveys NeSy approaches to visual query answering. The paper discusses the use of PL as a formal language for describing domains and the encoding of PL formulas as vectors in a GCNN ([Berlot-Attwell 2021](#), p. 9, Section 4). NeSy reasoning capabilities and proving or assurance levels in PL are not discussed.

The paper [Garcez et al. \(2019\)](#) reviews recent advances in NeSy computing, which is an approach to integrating machine learning with reasoning. The authors use a four-dimensional taxonomy to summarize NeSy techniques ([Garcez et al. 2019](#), p. 627, Table 5). The knowledge representation dimension (see ID 17 in Table 15) includes PL as a formal language integrated within four NeSy frameworks. For example, the authors describe how PL formulas are represented in neural networks such as KBANN [Towell and Shavlik \(1994\)](#), and CILP [d’Avila Garcez et al. \(2012\)](#). This topic has already been discussed in the survey ([Hamilton et al. 2022](#), Figures 28–34, p. 7).

The survey paper [Townsend et al. \(2019\)](#) does not directly employ PL. Instead, it adopts the Andrews–Diederich–Tickle (ADT) taxonomy of rule-extraction and rule-refinement techniques, originally proposed in 1995 ([Andrews et al. 1995](#), pp. 373–377). In the original work, this taxonomy classifies techniques that extract rules from trained ANNs or use ANNs to refine existing rule bases, including decompositional, pedagogical, and eclectic techniques ([Andrews et al. 1995](#), p. 376). Since the original taxonomy dates to the 1990s discussed in Section 5.3, the survey [Townsend et al. \(2019\)](#) is treated here as a later survey that borrows this earlier perspective to frame and compare NeSy AI approaches for extracting relational explanations from DNN.

A structured survey paper [Bader and Hitzler \(2005\)](#) reviews NeSy frameworks along three classification dimensions (see Table 15). Along the Language dimension, PL is discussed as a knowledge representation language that can be embedded into neural networks. KBANN [Towell and Shavlik \(1994\)](#) is cited as a concrete example of encoding PL formulae within a neural architecture. Under the Usage dimension, the authors discuss reasoning in PL within NeSy frameworks, but theorem-proving capabilities and assurance levels in PL are not addressed.

The paper [Bouneffouf and Aggarwal \(2022\)](#) provides a comprehensive review of NeSy applications. In their survey, the authors adopt the three-dimensional taxonomy introduced in [Bader and Hitzler \(2005\)](#). PL is mentioned as one of the NeSy languages used in the domains covered. Nonetheless, the paper does not discuss the reasoning and theorem proving capabilities or the assurance levels in PL.

5 Literature Survey

This section surveys 181 research papers on NeSy frameworks [within the search period 1940–2026](#). Publications were collected using OpenAI’s web search tool with the prompt given in Appendix A.2. The prompt defines the publication periods, eligibility criteria, search terms, metadata requirements, and deduplication rules. The search covered publisher and conference platforms, arXiv, OpenReview, DBLP, Semantic Scholar, OpenAlex, Crossref, Scopus, Web of Science, IEEE Xplore, ACM Digital Library, Springer Nature Link, Springer Nature, institutional repositories, and authors’ publication pages. Each reference in the resulting BibTeX file was manually screened to verify its bibliographic metadata and assess its eligibility. References that did not meet the survey’s inclusion criteria were excluded. Notes were also added to the relevant bibliography entries to identify preprints and works for which peer review or acceptance for publication could not be confirmed (e.g., [Bünz and Lamm \(2017\)](#)). The publications are organized into nine chronological periods, comprising eight complete decades and the partial decade 2020–2026. Table 21 summarizes the number of papers surveyed in each period. No surveyed papers were published during 1950–1979. The tables in each subsection summarize the X (capability) and Y (assurance) annotations assigned to each paper and identify the ANN architectures used. For each study, the accompanying discussion examines its PL formula representation. Labels such as [RAN- \$k\$ -SAT Abubakar and Danrimi \(2021\)](#), [MAJ2SAT Alway et al. \(2022\)](#), and [CRAN2SAT Roslan et al. \(2024\)](#) are treated here as study-specific names for PL formula families or encodings, rather than as names of distinct standardized logical languages.

5.1 1940–1979

The association of PL with neural activation can be historically attributed to [McCulloch and Pitts \(1943\)](#), whose work provided a foundational basis for later NeSy frameworks. Motivated by the all-or-none character of neural activity, authors proved that activation of neurons can be treated using PL formula. Instead of learning, their strategy relies on rule-based manual encoding: networks of neurons were converted into PL formulas (X6.2: logic expressivity; see Table 22). In the recursive construction ([McCulloch and](#)

Table 21. Distribution of surveyed papers describing NeSy frameworks for inference, reasoning, and proving in PL across chronological periods.

Group	Years	Decade	Number of papers
G1	1940–1949	1940s	1
G2	1950–1959	1950s	0
G3	1960–1969	1960s	0
G4	1970–1979	1970s	0
G5	1980–1989	1980s	2
G6	1990–1999	1990s	24
G7	2000–2009	2000s	15
G8	2010–2019	2010s	39
G9	2020–2026	2020s	100
TOTAL:			181

Pitts 1943, p. 8, Theorem III), each neuron N is assigned a unique numeral i , and synaptic connections between neurons are represented using logical connectives. To simulate reasoning, truth tables are used to evaluate all possible input combinations, assigning each proposition a truth value (T or F) (X2.1: satisfiability witness generation). This work does not integrate proving capabilities with ANN.

Table 22. X–Y annotations and ANN architectures of NeSy frameworks, 1940–1979.

ID	Reference	X and Y concept	ANN architecture
1	McCulloch and Pitts (1943)	X2.1, X6.2	McCulloch–Pitts ANN

5.2 1980–1989

With no papers identified for 1950–1979, the rule-based McCulloch–Pitts network (1940–1979) McCulloch and Pitts (1943) gives way to BPNNs and analog ANNs (1980–1989). Satisfiability witness generation (X2.1) and logic expressivity (X6.2) persist. The next decade, 1980–1989, adds logical consequence (X1.2), heuristic adaptation (X3.1.2), search-space pruning (X4.3), and SAT solving (X3.1.1.1). Neither decade reports Y concepts. Table 23 lists the X–Y–Z annotations and ANN architectures of the publications surveyed for 1980–1989.

Ertel and co-authors describe neural-guided PL theorem proving (X6.2: logic expressivity) Ertel et al. (1989). A symbolic PL theorem prover performs proof, while a BPNN learns global and local heuristics to reduce search space (X4.3: search-space pruning) (Ertel et al. 1989, p. 88, Figure 1). Thus, the

framework supports PL logical consequence (X1.2: logical consequence) via heuristic adaptation (X3.1.2: heuristic adaptation).

Table 23. X–Y annotations and ANN architectures of NeSy frameworks, 1980–1989.

ID	Reference	X and Y concept	ANN architecture
1	Ertel et al. (1989)	X3.1.2, X4.3, X1.2, X6.2	BPNN
2	Johnson (1989)	X2.1, X3.1.1.1, X6.2	Analog ANN

The 3-satisfiability problem is defined as a SAT problem in PL where formulas are expressed in CNF (X6.2: logic expressivity) (Johnson 1989, Section 4). Circuit interconnections represent a particular SAT problem instance (Johnson 1989, Abstract). An analog ANN is therefore employed for SAT solving (X3.1.1.1: SAT solving). Successful outputs provide satisfying truth assignments (X2.1: satisfiability witness generation).

These studies illustrate two enduring integration patterns such as neural guidance of symbolic PL proof search and neural SAT solving. They also clarify the distinction between capability and assurance. Both frameworks Johnson (1989) Ertel et al. (1989) perform reasoning and proving tasks in PL, but neither provides independently verifiable proof evidence.

5.3 1990–1999

This section surveys studies published between 1990 and 1999, with Table 24 presenting the X–Y–Z annotations and ANN architectures of the publications surveyed for this period. The 1980s studies used BPNNs and analog ANNs, whereas the 1990s studies covered a wider range of architectures, including HNNs, BMs, KBANNs, FNNs, and RNNs. Studies from both decades report logical consequence (X1.2), satisfiability witness generation (X2.1), SAT solving (X3.1.1.1), and logic expressivity (X6.2). In this comparison, heuristic optimization for SAT (X4.5), and algorithmic alignment (X6.1) are used to annotate publications from the 1990s studies. The same applies to soundness guarantees (X7) and soundness assessment (Y3). By contrast, heuristic adaptation (X3.1.2) and search-space pruning (X4.3) are annotated only in the 1980s studies. These differences reflect the coverage of the surveyed literature. They do not establish when a capability or architecture first emerged or whether it ceased to be used in the following decade.

In Monfreglio (1991), a ConnectNN for satisfying logical constraints is described. Since several logical constraint satisfaction problems can be reduced to SAT in PL, the paper focuses on solving SAT for formulas expressed in CNF using a simulated ANN trained through supervised learning

(Monfroglio 1991, pp. 29–49). Thus, the framework supports satisfiability witness generation (X2.1), and SAT solving (X3.1.1.1).

Table 24. X–Y annotations and ANN architectures of NeSy frameworks, 1990–1999.

ID	Reference	X and Y concept	ANN architecture
1	Monfroglio (1991)	X2.1, X3.1.1.1	ConnectNN
2	Pinkas (1991b)	X2.1	ConnectNN, SNN
3	Pinkas (1991a)	X2.1	SNN
4	Wan Abdullah (1992)	X2.1, X6.2	HNN
	Wan Abdullah (1996)		
5	Narazaki and Ralescu (1992)	X6.1, X6.2	motivated by HNN
6	Towell and Shavlik (1994)	X1.2	KBANN
	Towell et al. (1990)		
7	Wan Abdullah (1993)	X2.1, X6.2	HNN
8	d’Anjou et al. (1993)	X2.1, X3.1.1.1, X6.2	BM
9	Monfroglio (1993)	X3.1.1.1, X6.2	ConnectNN
	Monfroglio (1994)		
	Monfroglio (1995)		
10	Tambouratzis (1995)	X3.1.1.1, X6.2	SAANN
11	Pinkas (1995)	X6.2	ConnectNN
12	Weitze and Hofacker (1995)	X6.2	FNN
13	Nagamatu and Yanaru (1996)	X3.1.1.1, X2.1,	LPPH
		X6.1, X6.2, X7	
14	Roth (1996)	X1.2, X6.2, X7, Y3	ConnectNN
15	Spears (1996)	X2.1, X3.1.1.1, X6.2	HNN
16	Ritz and Hofacker (1996)	X6.1, X6.2	FNN
17	Alberti et al. (1997)	X3.2	HNN
18	Monfroglio (1999)	X3.1.1.1, X6.2, X4.5	RNN
19	d’Avila Garcez et al. (1999)	X6.2, X7, Y3	FNN
	Garcez and Zaverucha (1999)		

The theoretical work in Pinkas (1991b) Pinkas (1991a) sets out a formal way to combine PL with an SNN to solve the SAT problem in PL (X2.1: satisfiability witness generation; see Table 24). The proposed framework systematically transforms any well-formed PL formula written in CNF into a quadratic energy function. Furthermore, the author shows that any quadratic energy function can be used to represent a well-formed PL formula, thereby

establishing a connection between PL and a ConnectNN. Figure 1 in Pinkas (1991b) demonstrates how a ConnectNN with symmetric weights is constructed from a well-formed PL formula. The work presented in Pinkas (1991b) is chiefly theoretical and does not include empirical benchmarking. For instance, the study evaluates neither the ConnectNN's ability to process UNSAT PL formulas nor its scalability to PL formulas as the number of propositional variables increases. A subsequent study Tran and Garcez (2023) has criticized the scalability and representational complexity of the symmetric ANN approach. A follow-up paper Pinkas (1995) expands upon the work in Pinkas (1991b) by integrating a form of PL called penalty logic (Pinkas 1995, p. 206, Section 2) into a ConnectNN (X6.2: logic expressivity). When successful, the ConnectNN finds a truth assignment that satisfies the input PL formula (X2.1: satisfiability witness generation). Finding such an assignment does not establish that the PL formula is true under all truth assignments to its propositional variables Pinkas (1991b).

In Wan Abdullah (1992, 1996), the author implements Horn clauses (X6.2: logic expressivity) in a HNN with multi-connections. Connection strengths are defined from the logic program, and network relaxation optimizes logical consistency. When the network reaches a zero-cost state, the resulting truth assignment satisfies all clauses of the logic program (X2.1: satisfiability witness generation) (Wan Abdullah 1992, Section III, pp. 515–517).

An NN-like inference machine inspired by the HNN model is proposed in (Narazaki and Ralescu 1992, p. 746, see Figure 1). Nodes in the network represent propositions (X6.2: logic expressivity). The inference mechanism is implemented as a process of minimizing an energy function using the relaxation method to search for a truth-value distribution that achieves optimal consistency with the given data (X6.1: algorithmic alignment) (Narazaki and Ralescu 1992, see Abstract).

A KBANN Towell et al. (1990) is one of the earliest NeSy approaches in the 90s for integrating PL with trainable ANN (Towell and Shavlik 1994, see Abstract). Rules expressed as Horn clauses are compiled into an ANN architecture via a seven-step rules-to-network translation process: rewriting, mapping, numbering, adding hidden units, adding input units, adding links, perturbing (Towell and Shavlik 1994, Section 3.4). The resulting network is trained using the backpropagation algorithm (Towell and Shavlik 1994, p. 5, Section 3.1). The KBANN performs deductive reasoning (X1.2: logical consequence; see Table 24) based on embedded rules but does not perform automated theorem proving. The KBANN is evaluated on DNA sequence analysis (Towell and Shavlik 1994, p. 13, Section 4.2; p. 14, Section 4.3). Empirical tests indicate that the KBANN is an effective classifier

by comparison to other methods examined by authors (Towell and Shavlik 1994, p. 15, Section 5).

Logic programs are translated into a PL formula (X6.2: logic expressivity). The formula is then implemented in an HNN to minimize clause violations under the resulting interpretation of the clauses or logic programs (Wan Abdullah 1993, p. 203). The cost function is written as a sum of terms. Each term represents the negation of a clause in the logic program. The underlying task of the program is to find an assignment of truth values to the propositional variables such that all the clauses are satisfied (X2.1: satisfiability witness generation) (Wan Abdullah 1993, p. 203, Second paragraph).

Boltzmann machines (BM) for solving the SAT problem in PL are proposed in (d'Anjou et al. 1993, p. 514). SAT instances are encoded in a BM's consensus function (X6.2: logic expressivity). Simulated annealing uses this function to search for a satisfying configuration (X3.1.1.1: SAT solving). If the search finds a satisfying configuration, it produces a satisfiability witness generation (X2.1). Failure to find a satisfying configuration does not prove UNSAT. No assurance capabilities from the Y taxonomy are reported for this framework d'Anjou et al. (1993).

The CSP is expressed as a SAT problem in PL (X6.2: logic expressivity), with formulas written in CNF. The problem is encoded in several classes of ConnectNN (Monfreglio 1993, p. 175, Section 3). These ConnectNNs are trained to solve the CSP using a supervised procedure (X3.1.1.1: SAT solving) (Monfreglio 1993, p. 175, Section 3). In later research, the author uses backpropagation networks to solve the CSP (Monfreglio 1994, see Abstract). Subsequently, several ANNs are also trained using a supervised procedure that includes backpropagation learning (Monfreglio 1995, see Abstract) to solve the SAT problem in PL, with formulas written in CNF (X3.1.1.1: SAT solving; X6.2: logic expressivity). In Monfreglio (1999), the same author addresses the SAT problem in PL for PL formulas in CNF (X3.1.1.1: SAT solving; X6.2: logic expressivity). An RNN is used to select simplex pivot positions and improve the algorithm's performance (X4.5: heuristic optimization for SAT) (Monfreglio 1999, Abstract).

In Tambouratzis (1995), the author constructed an SAANN based on Harmony theory for solving the SAT problem in PL (X6.2: logic expressivity) (Tambouratzis 1995, p. 340, Abstract). The author describes the task as finding at least one clause which can be satisfied by appropriate values of the variables (X3.1.1.1: SAT solving) (Tambouratzis 1995, p. 341, Section 3). None of the assurance concepts in the Y dimension apply to this framework.

In Weitze and Hofacker (1995), the authors describe how they implement PL as a Boolean function (X6.2: logic expressivity) in ANNs. The

authors use two types of ANN: one type has two feedforward layers, and the other is a fully backcoupled network with N units (Weitze and Hofacker 1995, p. 181, Section 3). This work does not discuss how reasoning or proving in PL can be performed by neural networks.

LPPH is applied to a SAT problem in PL (X6.2: logic expressivity) Nagamatu and Yanaru (1996). The authors prove that SAT solutions are equilibrium points of the LPPH and vice versa (X7: soundness guarantee). Encoding SAT solutions as equilibria supports SAT solving (X3.1.1.1), satisfiability witness generation (X2.1), and algorithmic alignment (X6.1). Complete UNSAT refutation is not established.

Roth presents a connectionist architecture (ConnectNN) based on Valiant's Neuroidal model (Roth 1996, pp. 1256–1258; Abstract). Deduction is treated as deciding whether a Boolean function, described as a PL expression, entails a query (X1.2: logical consequence), with queries represented as conjunctions of rules whose antecedents and consequents are conjunctions of PL literals (X6.2: logic expressivity) (Roth 1996, pp. 1257–1260). Theorem 3 gives conditions under which deduction is correct (X7: soundness guarantee; Y3: soundness assessment) (Roth 1996, p. 1260).

Spears presents a neural network algorithm (NNSAT), an HNN algorithm for solving the SAT problem in PL (X3.1.1.1: SAT solving) (Spears 1996, p. 1121). The neural network graph is based on the AND/OR parse tree of a PL expression (X6.2: logic expressivity) (Spears 1996, pp. 1121–1122, Section 2). Figure 3 specifies that the NNSAT returns a satisfying truth assignment for the PL expression, if found (X2.1: satisfiability witness generation) (Spears 1996, p. 1124).

Ritz and Hofacker (1996) present an explicitly constructed FNN for computing the truth value of PL formulas with n variables written in complete DNF (X6.2: logic expressivity) (Ritz and Hofacker 1996, p. 553, see Figure 2 for an example). Ritz and Hofacker (1996) also construct a Turing machine whose tape output is the truth value of the input complete DNF (Ritz and Hofacker 1996, pp. 550–551, Sections 4.1–4.2). In Section 6, the authors report that FNNs with one hidden layer are trained using the standard backpropagation algorithm to compute the truth value of the PL formula in complete DNF (X6.1: algorithmic alignment) (Ritz and Hofacker 1996, p. 553, Section 6).

A combinatorial optimization problem, called MAX-2SAT, is described in Alberti et al. (1997). This optimization problem consists of finding a truth assignment that maximizes the number of satisfied clauses containing two literals in PL (X3.2: MaxSAT optimization). NAMS is a neural approximation algorithm for solving the MAX-2SAT problem (Alberti et al. 1997, Section 3). It uses an HNN to approximately maximize the number of satisfied 2-clauses.

Satisfying every clause is not guaranteed. The Y annotation set remains empty because no qualifying assurance mechanism was identified in the study.

One NeSy approach involves translating a PL program, expressed as general clauses (d'Avila Garcez et al. 1999, p. 5, Definition 1) (X6.2: logic expressivity; see Table 24), into an FNN (d'Avila Garcez et al. 1999, Section 2). This method can translate any PL program into an FNN with one hidden layer, as formally proved in Theorem 3 of the same work. This translation procedure is sound (X7: soundness guarantee; Y3: soundness assessment) d'Avila Garcez et al. (1999) and allows the resulting FNN to be trained using both example data and symbolic knowledge (d'Avila Garcez et al. 1999, Figure 1). Examples 4 and 9 illustrate how individual clauses map to neurons and connections in the FNN (d'Avila Garcez et al. 1999, p. 12, Example 4; p. 15, Example 9). The resulting FNN is depicted in Figure 3 of the same work (d'Avila Garcez et al. 1999, p. 13, Figure 3). This approach, implemented in the connectionist system $C - IL^2P$ Garcez and Zaverucha (1999), has been applied to tasks such as DNA sequence classification. In this application, the training set consisted of 3,190 DNA sequences, each containing 60 nucleotides. The performance of the $C - IL^2P$ system was evaluated using a cross-validation methodology, and the system was found to outperform KBANN Towell and Shavlik (1994).

5.4 2000–2009

This section surveys literature published between 2000 and 2009 and summarizes the X-Y annotations and ANN architectures of the corresponding NeSy frameworks in Table 25. The surveyed studies from both decades, 1990s and 2000s, use HNNs, FNNs, and LPPH. The 2000s studies also include controlled LPPH and Little HNN variants. Studies from both decades report logical consequence (X1.2), satisfiability witness generation (X2.1), SAT solving (X3.1.1.1), heuristic optimization for SAT (X4.5), algorithmic alignment (X6.1), logic expressivity (X6.2), and soundness guarantees (X7). Resolution (X3.1.1.2), strategy adaptation (X3.1.2.4), and branching heuristics (X4.4) are annotated in the 2000s. Soundness assessment (Y3) is annotated in the 1990s.

The LPPH dynamics use coefficients of attention to find Boolean assignments satisfying PL formulas expressed in CNF (X6.2: logic expressivity) (Nagamatu et al. 2000, pp. 65–66, Figure 1). The LPPH-CA model Nagamatu et al. (2000), which extends the LPPH framework theoretically analyzed in Nagamatu and Yanaru (1996), introduces dynamically controlled coefficients of attention for PL clauses. Neurons

represent PL variables, and polarized high-order connections represent clauses (Nagamatu et al. 2000, p. 66, Figure 1). The aforementioned coefficients of attention prioritize difficult clauses, supporting satisfiability witness generation (X2.1), SAT solving (X3.1.1.1), and heuristic optimization for SAT (X4.5). Experimental results demonstrate that these coefficients enhance LPPH performance, especially when addressing difficult SAT problems (Nagamatu et al. 2000, Sections 5–7). However, since the equilibrium theorem is stated for the original LPPH dynamics rather than the modified attention-controlled dynamics, no Y concept is assigned (Nagamatu et al. 2000, p. 67, Theorem 1).

ARQ-PROP-II is a propositional neural reasoner that finds proofs by refutation using the resolution principle (X3.1.1.2: resolution; X6.2: logic expressivity) (Lima et al. 2008, Abstract; p. 1037, Figure 1) (Lima 2001, see Abstract). The neural mechanism presented in the paper does not require the knowledge base to be pre-encoded or learned (Lima 2001, see Abstract). Vilela and Lima (2001) examine the cognitive plausibility of ARQ-PROP-II previously described in Lima (2001). The neural theorem prover produces proofs by refutation with limited depth in PL using resolution (X3.1.1.2: resolution) including monotonic reasoning (Vilela and Lima 2001, p. 823, Section 2). Consequently, all three papers Lima (2001); Vilela and Lima (2001); Lima et al. (2008) receive no annotation from the Y dimension.

An information-based annealing neural network approach (INN) is applied to solve combinatorial CSP Jönsson and Söderberg (2001). In this work, a CSP is reduced to the SAT problem for a PL formula written in CNF (hereafter, SAT for CNF formulas) (X3.1.1.1: SAT solving) Jönsson and Söderberg (2001). Successful runs return a truth assignment that satisfies the PL formula written in CNF (X2.1: satisfiability witness generation) (Jönsson and Söderberg 2001, Figure 2, p. 6). Failure does not establish unsatisfiability (Jönsson and Söderberg 2001, Section 2, p. 1). An empty Y annotation set indicates that no qualifying assurance mechanism was identified in the reported evidence; it does not establish that the framework is unsound.

An HT ANN tests SAT in PL (X3.1.1.1: SAT solving), converting PL formulas to DNF before testing (X6.2: logic expressivity). It treats SAT as a constraint optimization problem, using simulated annealing and a harmony function to evaluate truth assignments (X4.5: heuristic optimization for SAT). The abstract claims that the tester determines satisfiability, returns assignments maximizing the number of satisfied clauses, and can enumerate all maximizing assignments, yielding satisfying witnesses when possible (X2.1: satisfiability witness generation) Tambouratzis (2001).

Hitzler et al. (2004) define (normal) logic programs as clauses

Table 25. X–Y annotations and ANN architectures of NeSy frameworks, 2000-2009.

ID	Reference	X and Y concept	ANN architecture
1	Nagamatu et al. (2000)	X2.1, X3.1.1.1	LPPH
2	Jönsson and Söderberg (2001)	X4.5, X6.2 X2.1, X3.1.1.1	INN
3	Lima (2001) Vilela and Lima (2001) Lima et al. (2008)	X3.1.1.2, X6.2	higher order HNN
4	Tambouratzis (2001)	X2.1, X3.1.1.1, X4.5, X6.2	HT ANN
5	Hitzler et al. (2004)	X1.2, X6.1,	FNN , ConnectNN
6	Seda (2006) Zhang and Nagamatu (2005a) Zhang and Nagamatu (2005b)	X6.2, X7 X2.1, X3.1.1.1 X4.5, X6.2	LPPH
7	Kibria (2007)	X1.1, X2.1, X3.1.1.1 X3.1.2.4, X4.4, X4.5, X6.1	FNN
8	Sathasivam (2008) Sathasivam and Wan Abdullah (2008)	X2.1, X6.1 X6.2 , X3.1.1.1	HNN
9	Sathasivam (2009a) Sathasivam (2009b)	X3.1.1.1, X2.1 X6.2	Little HNN

$$A \leftarrow L_1 \wedge \cdots \wedge L_n \quad (9)$$

and show that, for each PL program P , a 3-layer FNN of binary threshold units computes the immediate consequence operator T_P (X1.2: logical consequence; X6.1: algorithmic alignment; X6.2: logic expressivity) (Hitzler et al. 2004, p. 254, Theorem 3.2). For strongly determined PL programs, the corresponding recurrent network converges to the unique supported model (X7: soundness guarantee) (Hitzler et al. 2004, p. 256, Corollary 3.3). Additional work integrating logic programs with ConnectNN has been published in Hitzler et al. (2004); Seda (2006). In one such approach, Hitzler et al. (2004) apply Funahashi’s theorem (Funahashi 1989, p. 184, Theorem 1). Consequently, one can construct a neural network that computes the immediate consequence operator for each PL program (Seda 2006, p. 117, Theorem 2.3, Section 2.3).

LPPH is proposed for solving the SAT problem in PL (X3.1.1.1: SAT solving; X6.2: logic expressivity) (Zhang and Nagamatu 2005a, pp. 969–970,

Sections 1–2) (Zhang and Nagamatu 2005b, p. 121, Section 2). During parallel execution, it finds a truth assignment for a PL formula expressed in CNF (X2.1: satisfiability witness generation). By using mixed biases during parallel execution, LPPH supports heuristic optimization for SAT (X4.5) (Zhang and Nagamatu 2005b, p. 122, Section 4).

Kibria (2007) modifies MINISAT to use the neural net-based heuristic implemented with a multilayer FNN (X6.1: algorithmic alignment) (Kibria 2007, Section II-A). The underlying DPLL algorithm operates on CNFs and determines whether a problem is SAT or UNSAT (X1.1: satisfiability decision), finding a model if the problem is satisfiable (X2.1: satisfiability witness generation; X3.1.1.1: SAT solving) (Kibria 2007, Sections I-A–I-B). The network receives parameters relating to the search and adjusts variable activity increments. The decision function selects the variable with the highest activity. A further network output determines the value to assign when the variable is chosen again (X4.4: branching heuristic). The network also increments counters that control restarts and clause database reduction (X3.1.2.4: strategy adaptation) (Kibria 2007, Section II-A). An evolution strategy (ES) was employed to determine the neural network’s weights and biases, as well as the values of the constants (X4.5: heuristic optimization for SAT) (Kibria 2007, Section II-B).

Hebbian and Wan Abdullah’s learning methods in an HNN are compared Sathasivam (2008); Sathasivam and Wan Abdullah (2008). Horn clauses are translated into Boolean algebraic form (X6.2: logic expressivity) (Sathasivam 2008, see Formula 5). The network finds a truth assignment that satisfies every clause and therefore satisfies the PL formula written in CNF (X3.1.1.1: SAT solving; X2.1: Satisfiability witness generation) (Sathasivam 2008, Section 2, p. 323; Section 4, p. 325). Wan Abdullah’s method obtains the values of connection strengths by comparing the cost function with the energy (X6.1: Algorithmic alignment) (Sathasivam 2008, Section 4, steps (iv)–(vi), p. 325) (Sathasivam and Wan Abdullah 2008, Sections 2–4). The same learning methods are compared experimentally in (Sathasivam and Wan Abdullah 2008, Sections 4–5), using stable states and their energies to assess whether both methods yield similar results for the given clauses.

In Sathasivam (2009a) and Sathasivam (2009b), the author uses a Little HNN to encode propositional program clauses expressed in CNF (X6.2: logic expressivity). Energy relaxation seeks interpretations satisfying all clauses (X3.1.1.1: SAT solving; X2.1: satisfiability witness generation) (Sathasivam 2009a, Section 3) (Sathasivam 2009b, p. 162, Section III). Storkey’s new learning rule is compared with the Hebb rule, with shorter relaxation times reported in (Sathasivam 2009a, Table 1). The sign constraint method yields

a higher reported global minima ratio than the original method (Sathasivam 2009b, p. 163, Table 1).

5.5 2010–2019

Surveyed 2000s studies use HNNs, FNNs and LPPH. However, 2010s studies retain HNNs and additionally include RBFNNs, DNNs, GNNs, MPNNs, and SATNet, as documented alongside their X–Y annotations in Table 26. Both decades report X1.1, X1.2, X2.1, X3.1.1.1, X4.5, X6.1, X6.2, and X7. Additional 2010s annotations include X3.1.2.2, MaxSAT optimization (X3.2), search-space pruning (X4.3), logical generalization (X5.1), robustness (X8.1), interpretability (X8.2), proof validation (Y1.2) and soundness assessment (Y3). Resolution (X3.1.1.2), strategy adaptation (X3.1.2.4) are annotated in the surveyed 2000s studies.

Abdechiri and Meybodi (2012) propose the HNNICA model, which combines an HNN with the ICA to solve the SAT problem in PL when a PL formula is expressed in CNF (X3.1.1.1: SAT solving; X6.2: logic expressivity). The SAT problem is reformulated using a Lagrange function (Abdechiri and Meybodi 2012, p. 729, Step 1). The HNN performs a local search (X4.3: search-space pruning), and the ICA performs a global search (Abdechiri and Meybodi 2012, p. 728, Section III; p. 729, Section IV.B).

In (Flint and Blaschko 2012, pp. 2771–2778), the authors represent PL formulas in CNF (X6.2: logic expressivity). Using modified perceptron-style updates, the priority function scores candidate branching literals from their feature representations and selects the highest-scoring literal for DPLL branching (X6.1: algorithmic alignment) (Flint and Blaschko 2012, Sections 2.1–3.2, Eqs. (1) and (3)). The DPLL algorithm determines whether a CNF sentence is satisfiable (X1.1: satisfiability decision; X3.1.1.1: SAT solving). The paper reports none of the assurance mechanisms represented in the Y dimension.

A GNN can be used to learn complete solvers for CSPs, particularly Boolean SAT (X3.1.1.1: SAT solving) in PL (Amizadeh et al. 2019, see Abstract). The authors created a design pattern that allows SAT problems to be solved using a GNN. This design pattern is a generalization of the general message-passing (GMP) algorithm (Amizadeh et al. 2019, Section 4). The authors used an unsupervised training strategy for the proposed solver, and the CNF clauses ranged in size from 2 to 10 (Amizadeh et al. 2019, Section 6.3).

A message-passing GNN is trained with a modified attention mechanism to guide branching in DPLL and CDCL algorithms (X4.4: branching heuristic; X4.5: heuristic optimization for SAT) (Jaszczur et al. (2019)). These algorithms determine the satisfiability of a PL formula written in CNF (X1.1: satisfiability decision; X3.1.1.1: SAT solving) (Jaszczur et al. 2019, p. 1, Algorithm 1).

Table 26. X–Y annotations and ANN architectures of NeSy frameworks, 2010-2019.

ID	Reference	X and Y concept	ANN architecture
1	Abdechiri and Meybodi (2012)	X3.1.1.1, X4.3, X6.2	HNN
2	Flint and Blaschko (2012)	X1.1, X3.1.1.1 X6.1, X6.2	Not given
3	Molnár and Ercsey-Ravasz (2013)	X2.1, X3.1.1.1 X6.1, X6.2	Cont. time ANN
4	França et al. (2014)	X2.1, X3.1.1.1 X1.2, X5.1 X8.2, Y3	ANN as directed graph
5	Hamadneh et al. (2014a) Hamadneh et al. (2014b)	X1.2, X6.1 X6.2	RBFNN
6	Sathasivam and Velavan (2014) Velavan et al. (2015)	X2.1 X6.1, X6.2	HNN
7	Bin Mohd Kasihmuddin et al. (2016)	X3.2, X4.5 X6.2	HNN
8	Kasihmuddin and Sathasivam (2016) Kasihmuddin et al. (2016a) Kasihmuddin et al. (2016b)	X2.1, X3.2 X6.1, X6.2	HNN
9	Mansor and Sathasivam (2016a) Mansor et al. (2016a) Mansor and Sathasivam (2016b) Mansor et al. (2016b)	X2.1, X4.5, X6.1 X6.2	HNN
10	Rocktäschel and Riedel (2017)	X7, X6.2	ANN for end-to-end diff. proving
11	Velavan et al. (2016)	X3.1.1.1, X4.5 X6.1, X6.2	HNN
12	Bünz and Lamm (2017)	X3.1.1.1	GNN
13	Mansor et al. (2017a) Mansor et al. (2017b)	X3.1.1.1, X3.2, X2.1 X4.5, X6.2	HNN
14	Alzaeemi and Sathasivam (2018)	X6.1, X6.2, X8.1	HNN
15	Tran (2018) Tran (2017)	X1.2, X3.1.1.1 X6.2	UANN
16	Mohd Kasihmuddin et al. (2018)	X3.2, X4.5 X6.2, X8.1	HNN
17	Sekiyama and Suenaga (2018)	X1.2, X5.1 X6.1, X6.2 X7, Y1.2, Y3	DNN
18	Wang and Rompf (2018)	X3.1.1.1, X4.4	ConvNN
19	Kasihmuddin et al. (2019)	X6.1, X6.2	HNN
20	Kathirvel et al. (2019)	X3.1.1.1, X6.2, X8.1	HNN
21	Kho et al. (2019)	X3.1.1.1, X4.5 X6.2, X8.1	HNN
22	Mohd Kasihmuddin et al. (2019)	X3.1.1.1, X6.2	HNN
23	Evans et al. (2018)	X1.2	Possible WorldNet multi-layer
24	Amizadeh et al. (2018)	X3.1.1.1	
25	Selsam et al. (2019)	X3.1.1.1	MPNN
26	Yolcu and Póczos (2019)	X3.1.1.1, X4.5	GNN
27	Wang et al. (2019)	X3.1.1.1, X3.2	SATNet, ConvNN
28	Jaszczur et al. (2019)	X1.1, X3.1.1.1 X4.4, X4.5	GNN
29	Amizadeh et al. (2019)	X3.1.1.1	GNN
30	Mansor et al. (2019)	X6.2, X3.1.1.1	Elliot HNN

In (Molnár and Ercsey-Ravasz 2013, p. 2), the authors encode a PL formula expressed in CNF as a matrix that determines connections in an asymmetric continuous time ANN (X6.1: algorithmic alignment; X6.2: logic expressivity). The dynamics of the ANN search for a truth assignment that satisfies every clause (X2.1: satisfiability witness generation; X3.1.1.1: SAT solving).

The $C\text{-IL}^2P$ system uses recurrent rule chaining to compute consequences of the encoded PL program (X1.2: logical consequence). After training, it classifies previously unseen examples (X5.1: logical generalization) and can

extract a revised propositional theory (X8.2: interpretability) (França et al. 2014, pp. 86–88, Section 2.2). The rule extraction procedure is reported to be sound and complete for regular networks and sound in the general case (Y3: soundness assessment) (França et al. 2014, p. 88, Section 2.2). CILP++ is a system for relational learning França et al. (2014). It is a successor to the $C - IL^2P$ system. CILP++ uses the propositionalization method called bottom clause propositionalization (BCP) (França et al. 2014, p. 88, Section 2.3). BCP transforms relational data into so-called propositional feature representations, yielding an attribute-value table suitable for ANN learning. CILP++ performs neural decision-based inference (X1.2: logical consequence; see Table 26) on unseen data and learns from relational data. The CILP++ system is not designed to perform theorem proving tasks in PL.

A PL program is a finite set of PL clauses (Hamadneh et al. 2014a, see Abstract). The PL program is represented in an RBFNN by computing the single-step operators of logic programming (X1.2: logical consequence). The clause body literals form the inputs of the RBFNN, and the clause heads form the outputs (X6.1: algorithmic alignment; X6.2: logic expressivity) (Hamadneh et al. 2014a, p. 462, Figure 1). A PL program is in 3-CNF if each clause in CNF consists of a maximum of three literals (Hamadneh et al. 2014b, p. 548). Related work (Hamadneh et al. 2014b, p. 548) also represents PL programs in RBFNN but proposes satisfiability testing for 3-CNF, with outputs representing clauses rather than clause heads, using the prey-predator algorithm to calculate the output weights of the RBFNN.

Sathasivam and Velavan (2014) represent propositional Horn clauses in a higher-order HNN (X6.2: logic expressivity). Neurons store atoms' truth values, and the cost function is minimized when all clauses are satisfied (X2.1: satisfiability witness generation; X6.1: algorithmic alignment) (Sathasivam and Velavan 2014, p. 140, Section 1). When relaxation of the HNN reaches a global energy minimum, the neuron states provide an interpretation satisfying all clauses (Sathasivam and Velavan 2014, p. 142, Section 3). Velavan et al. (2015) substantially overlaps with this work regarding Horn clause encoding in higher-order HNN. However, it adds agent-based modeling in NetLogo (Velavan et al. 2015, Section V). In contrast, Sathasivam and Velavan (2014) uses C++ simulations.

Integrating a genetic algorithm with an HNN can maximize the number of satisfied clauses in a PL formula expressed in CNF (X3.2: MaxSAT optimization; X4.5: heuristic optimization for SAT; X6.2: logic expressivity) (Bin Mohd Kasihmuddin et al. 2016, pp. 52–54). Candidate truth assignments are evaluated, selected, recombined, and mutated. The NeSy framework described in Bin Mohd Kasihmuddin et al. (2016) does not establish UNSAT or generate a formal proof. Moreover, the paper does not describe

any procedures for verification, certification, proof validation, interactive explanation, or soundness assessment.

Three studies encode clauses in HNNs (X6.2: logic expressivity), derive weights using Wan Abdullah’s method, and find assignments via energy relaxation (X6.1: algorithmic alignment) Kasihmuddin and Sathasivam (2016) Kasihmuddin et al. (2016a) Kasihmuddin et al. (2016b). In Kasihmuddin and Sathasivam (2016), authors compare unipolar and hyperbolic activation functions for finding satisfying interpretations (X2.1: satisfiability witness generation). However, (Kasihmuddin et al. 2016a, see Abstract) integrate the artificial bee colony search for solving restricted MAX- k -SAT (X3.2: MaxSAT optimization), while Kasihmuddin et al. (2016b) represent Bezier curve properties as PL formulas expressed in CNF, with at most two literals per clause (X6.2: logic expressivity) (Kasihmuddin et al. 2016b, Section II.A, p. 10) and use exhaustive search to find satisfying assignments (X2.1: satisfiability witness generation).

In these four studies, Mansor and co-authors translate PL clauses into Boolean algebra (X6.2: logic expressivity) and derive HNN parameters by matching the resulting cost function to network energy (X6.1: algorithmic alignment). An HNN state with a zero clause violation cost represents a satisfying interpretation and therefore supplies satisfiability witness generation (X2.1) (Mansor and Sathasivam 2016a, pp. 45, 46), (Mansor et al. 2016a, pp. 28 and 30), (Mansor and Sathasivam 2016b, pp. 2 and 3), (Mansor et al. 2016b, pp. 23 and 26). The activation function modifies relaxation to improve the frequency of finding satisfying assignments (X4.5: heuristic optimization for SAT) Mansor and Sathasivam (2016a,b).

Velavan et al. (2016) encode propositional Horn clauses in an HNN (X6.2: Logic expressivity), equating clause costs with network energy (X6.1: algorithmic alignment). Relaxation seeks satisfying interpretations (X3.1.1.1: SAT solving) (Velavan et al. 2016, p. 156, Section 3). Mean field theory modifies local field computation to improve model search (X4.5: Heuristic optimization for SAT) (Velavan et al. 2016, pp. 157–159, Sections 4 and 6).

A graph representation of PL formulas written in CNF is described in Bünz and Lamm (2017). A GNN is then trained to predict the satisfiability of these formulas (X3.1.1.1: SAT solving) (Bünz and Lamm 2017, Sections 4–6).

Inspired by the backward chaining algorithm, an ANN architecture called neural theorem prover (NTP) is constructed to perform end-to-end differentiable proving of queries to a knowledge base (Rocktäschel and Riedel 2017, see Abstract). In this NeSy framework, theorems are formulated as queries to a knowledge base, and proof search is implemented through rule application and unification (X7: soundness guarantee). As a result, the NTP can infer a new fact from an incomplete knowledge base. While the knowledge

base can be expressed in FOL, even though this is not explicitly stated in Rocktäschel and Riedel (2017), it is reasonable to note that facts can also be expressed in PL (X6.2: logic expressivity; see Table 26).

In Mansor et al. (2017a) and Mansor et al. (2017b), Mansor and co-authors implement an artificial immune system algorithm in an HNN to improve truth assignments (X4.5: Heuristic optimization for SAT). In (Mansor et al. 2017a, Abstract), the authors seek assignments satisfying all clauses (X2.1: satisfiability witness generation; X3.1.1.1: SAT solving) and evaluate global Hamming distance. Restricted MAX- k -SAT as a constraint optimization problem is solved by implementing the artificial immune system algorithm in the HNN (X3.2: MaxSAT optimization). The paradigm is compared with a Brute force search algorithm integrated in the HNN (Mansor et al. 2017b, Abstract). The MAX- k -SAT problem is defined in PL, with formulas expressed in CNF (X6.2: logic expressivity) (Mansor et al. 2017b, Section II).

Propositional Horn clauses in CNF are encoded into a Linear Kernel Hopfield ANN (X6.2: logic expressivity) (Alzaeemi and Sathasivam 2018, pp. 4 and 5). To initialize the HNN, all synaptic weights are set to zero. Synaptic connection strengths are obtained by comparing the energy function with the cost function (X6.1: algorithmic alignment). The robustness (X8.1) of the approach is supported by empirical results (Alzaeemi and Sathasivam 2018, Computation Time Section).

PL programs can be encoded in an RBM by translating them into the RBM's energy functions (X6.2: logic expressivity) Tran (2018). This encoding enables the RBM to carry out reasoning tasks, including solving SAT instances (X3.1.1.1: SAT solving; see Table 26). In a related line of work, the same author demonstrates how symbolic knowledge, expressed as if-then rules in the form of Horn clauses, can be embedded in an unsupervised ANN Tran (2017). In this setting, reasoning is implemented directly at the ANN level. Inference over relations between entities (X1.2: logical consequence) is performed by an algorithm that operates on the trained network rather than by an external solver (Tran 2017, p. 4, Algorithm 1). This illustrates a reasoning process in which symbolic reasoning emerges as an interpretation of neural inference, rather than being performed by an explicit symbolic reasoner (Tran 2017, p. 3, Section 4.2).

Limited MAX- k -SAT ($k = 2$ and $k = 3$) formulas expressed in CNF are mapped to a discrete HNN (X6.2: logic expressivity) (Mohd Kasihmuddin et al. 2018, pp. 1328–1331). The HNN is integrated with Wan Abdullah method and Sathasivam relaxation method to obtain the correct final state of the neurons (X3.2: MaxSAT optimization; X4.5: heuristic optimization for sat) (Mohd Kasihmuddin et al. 2018, see Abstract). Experimental results

emphasize robustness (X8.1) showing that the MAX- k -SAT can be embedded optimally in the HNN.

A DNN is used in automated proof synthesis for minimal propositional logic (X6.2: logic expressivity) (Sekiyama and Suenaga 2018, Section 2). The proposition-to-proof architecture is proposed, which is the DNN tailored to the automated proof synthesis problem (X6.1: algorithmic alignment) (Sekiyama and Suenaga 2018, Sections 3–4; see Abstract). This procedure generates proofs of propositions (X1.2: logical consequence), though it may fail to find a proof even for a provable proposition. Type checking validates the returned proofs and guarantees their correctness (X7: soundness guarantee; Y1.2: proof validation; Y3: soundness assessment) (Sekiyama and Suenaga 2018, Section 3.2, p. 316; Section 6.2, p. 326). Evaluating on test propositions provides evidence specific to the benchmark of logical generalization (X5.1) (Sekiyama and Suenaga 2018, Section 5.2, Table 2).

Casting branching decisions in a CDCL SAT solver as moves in a game enables the use of deep RL to guide these decisions (X3.1.1.1: SAT solving)(Wang and Rompf 2018, Sections1–2). PL formulas written in CNF are represented as sparse adjacency matrices. These matrices are supplied to a ConvNN used to learn branching heuristics (X4.4: branching heuristic)(Wang and Rompf 2018, Section2)

The research integrates PL formulas expressed in CNF into HNN (X6.1: algorithmic alignment; X6.2: logic expressivity) (Kasihmuddin et al. 2019, pp. 2–5, Sections 2–5). Propositional variables are mapped to neurons, and synaptic weights are obtained by using the Wan Abdullah method (Kasihmuddin et al. 2019, p. 1, Section 1; p. 7, Section 6). The authors explore the quality of the solution obtained from HNNs, which shows that HNN only retrieves a limited number of neuron states with the lowest minimum energy (Kasihmuddin et al. 2019, see Abstract). No certificate, external verification, or soundness assessment is provided.

The HNN-3SATICA model combines the ICA with an HNN to solve SAT problems in PL (X3.1.1.1: SAT solving), with formulas expressed in CNF (X6.2: logic expressivity) (Kathirvel et al. 2019, pp. 144–147). Each clause contains three literals. Empirical stability supports robustness (X8.1) (Kathirvel et al. 2019, Section 5).

2SAT denotes a formula in PL expressed in CNF, where each clause has two literals (X6.2: logic expressivity) (Kho et al. 2019, Section 2). The limitations of an HNN in finding a consistent interpretation that satisfies the formula P_{2SAT} are investigated (X3.1.1.1: SAT solving) Kho et al. (2019). Both ESM and ACO are utilized to solve the SAT problem (X4.5: heuristic optimization for SAT). Performance evaluation using error measures supports robustness (X8.1) (Kho et al. 2019, p. 5, Performance Evaluation Section).

An immune system algorithm is hybridized with an Elliot HNN to solve the SAT problem in PL, with formulas expressed in CNF (X6.2: logic expressivity; X3.1.1.1: SAT solving) Mansor et al. (2019). The main goal of this work is to investigate the effectiveness and capability of the proposed approach in relation to the number of neurons (Mansor et al. 2019, see Abstract).

k -SAT represents a PL formula in CNF where each clause has k literals (X6.2: logic expressivity) (Mohd Kasihmuddin et al. 2019, Section 2). k -SAT is integrated into a mutation HNN. The neuron states of the HNN converge to a minimum-energy state (Mohd Kasihmuddin et al. 2019, see Abstract). Figure 1 illustrates the implementation of propositional SAT in an HNN (X3.1.1.1: SAT solving) (Mohd Kasihmuddin et al. 2019, Section 2). A randomized k -SAT formula is provided as input. After clause satisfaction is checked, the randomized formula is stored in the HNN. Otherwise, following trial and error, falsified is reported (Mohd Kasihmuddin et al. 2019, p. 4, Figure 1).

In Evans et al. (2018), the authors investigate whether an ANN can understand logical formula well enough to detect entailment. To address this research question, the authors create a dataset consisting of triples that can be used to identify logical entailment in PL Evans et al. (2018). The authors evaluate ANN on this dataset and show how these ANNs can capture logical entailment (X1.2: logical consequence) Evans et al. (2018). The authors use the PossibleWorldNet ANN architecture, which outperforms all benchmarks, to approximate logical entailment. The paper compares this approach with other approaches and neural models (Evans et al. 2018, p. 8, Table 2).

In Amizadeh et al. (2018), the authors trained a multilayer ANN to solve the Circuit-SAT problem (X3.1.1.1: SAT solving; see Table 26) (Cormen et al. 2009, pp. 1070) (Trevisan 2001, pp. 2, Section 2, Definition 3). The input to the ANN is a PL formula, represented as a Boolean circuit. The proposed methodology for training the ANN involves using an ANN function to generate SAT solutions, even though the ANN is not exposed to such solutions during training (Amizadeh et al. 2018, pp. 6, Section 4.1). In related work, the NeuroSAT model Selsam et al. (2019) uses a message-passing ANN (MPNN) trained as a binary classifier with single-bit supervision to predict whether Boolean formulas in CNF are satisfiable ((X3.1.1.1: SAT solving); see Table 26). Each formula is encoded as a graph. NeuroSAT can also be applied to solve other combinatorial problems, such as graph coloring or vertex cover problems, by encoding them as SAT problem (Selsam et al. 2019, see Abstract).

The work Yolcu and Póczos (2019) used a GNN to solve SAT problem (X3.1.1.1: SAT solving). This approach differs from the work in Selsam et al.

(2019), which predicts SAT. The GNN is a part of a stochastic local search (SLS) algorithm. Here, the GNN acts as a learned variable-selection heuristic Yolcu and Póczos (2019). This heuristic is learned through a process called reinforcement learning (X4.5: Heuristic optimization for SAT; see Table 26) (Yolcu and Póczos 2019, pp. 5, Section 6.1). In the experiment, the results are compared with those of WalkSAT Selman et al. (1993) (Yolcu and Póczos 2019, pp. 7, Section 7.1) in terms of steps required to find a solution, with a maximum of 750 steps, and the percentages of PL formulas that are correctly solved (Yolcu and Póczos 2019, pp. 7, Table 2). For example, if one is trying to solve the SAT problem where the maximum number of variables is 50 and the maximum number of clauses is 213 (Yolcu and Póczos 2019, pp. 7, first row in Table 2), the percentage of formulas solved using the GNN approach is higher than that solved by the WalkSAT solver (Yolcu and Póczos 2019, pp. 7, second and last cell in the first row of Table 2).

A differentiable, approximate MaxSAT solver, called SATNet, is proposed in Wang et al. (2019). SATNet implements a differentiable approximation to the MaxSAT problem in PL (X3.1.1.1: SAT solving; X3.2: MaxSAT optimization; see Table 26) by relaxing the MaxSAT problem to a semidefinite program (SDP). The solver can be integrated with a larger ConvNN architecture, enabling end-to-end training via gradient-based learning (Wang et al. 2019, see Abstract and Section 4.3).

5.6 2020–2026

This section surveys studies published from 2020 to 2026, tracing their X–Y annotations and ANN architectures through annual summaries for 2020 (Table 27), 2021 (Table 28), 2022 (Table 29), 2023 (Table 30), and 2024 (Table 31), followed by a combined summary for 2025–2026 (Table 32). Surveyed studies from both 2010–2019 and 2020–2026 use HNNs, DNNs, GNNs, MPNNs and RBMs. The studies 2020–2026 also include LNNs, Transformers, Tree-LSTMs and HyperGCNs. All X annotations assigned to the surveyed 2010s studies, except robustness (X8.1), also appear in the surveyed 2020–2026 studies, including SAT solving (X3.1.1.1). Proof validation (Y1.2) and soundness assessment (Y3) also appear in both periods. Additional annotations in the surveyed 2020–2026 studies include UNSAT refutation (X2.2), resolution (X3.1.1.2), query-aware formula generation (X3.1.2.3), strategy adaptation (X3.1.2.4), clause selection quality (X4.2), branching heuristics (X4.4), UNSAT-core prediction (X5.3), certificate support (Y1.1), certificate format (Y1.1.1) and UNSAT certification (Y1.4.1).

The HNN-RAN- k -SAT study Sathasivam et al. (2020c) considers a specific Random k -satisfiability (RAN- k -SAT) family of PL formulas expressed in CNF (X6.2: logic expressivity). The authors encode these formulas in an HNN, with

synaptic weights obtained using the Wan Abdullah method Wan Abdullah (1992) (X3.1.1.1: SAT solving). They state that exhaustive search makes it difficult for HNN-RAN- k -SAT to learn what they describe as the *inconsistent interpretation*, $\neg P_{\text{RANKSAT}}$ (Sathasivam et al. 2020c, p. 2855).

E2SAT is a class of PL formulas expressed in CNF with exactly two literals per clause (X6.2: logic expressivity) (Sianipar et al. 2020, p. 040014-2). HNN-E2SATACO uses ACO to enhance the learning phase of an HNN for E2SAT logic programming. ACO searches for consistent interpretations of E2SAT clauses during learning (X4.5: heuristic optimization for SAT) (Sianipar et al. 2020, pp. 040014-4–040014-5).

Graph-Q-SAT uses a GNN to solve SAT for CNF formulas (X3.1.1.1: SAT solving) Kurin et al. (2020). The GNN guides a CDCL solver, which generates a proof of unsatisfiability (Kurin et al. 2020, Introduction section, 3rd paragraph). The GNN predicts Q-values for variable assignments (X4.4: branching heuristic) (Kurin et al. 2020, Sections 3.1–3.2). The SAT problem is encoded as a graph representation, similar to the approach in Selsam et al. (2019). Graph-Q-SAT enables the trained policy to select branching variables efficiently. It can reduce the number of iterations required to solve SAT problems by a factor of 2–3 (Kurin et al. 2020, p. 1, see Abstract).

NLocalSAT uses a GGC-NN followed by a two-layer perceptron to predict a candidate truth assignment for the variables of an input PL formula written in CNF (Zhang et al. 2020, Section 2.2). These predictions guide the initialization of variable truth assignments in a stochastic local search solver used to solve the SAT problem for this formula (X3.1.1.1: SAT solving; X4.5: heuristic optimization for SAT). When successful, the solver returns a truth assignment satisfying every clause and thus the formula (X2.1: satisfiability witness generation) (Zhang et al. 2020, Section 2.4, Algorithms 2–3).

The neural-network-guided SAT attack (NNgSAT) is proposed to examine the use of an MPNN to predict a satisfying assignment for a PL formula written in CNF that the actual SAT solver cannot solve (X6.2: logic expressivity; X3.1.1.1: SAT solving) (Azar et al. 2020, Section 1.1).

The RANKSAT logic representation is defined as a class of PL formulas written in CNF with one or two literals per clause (X6.2: logic expressivity) (Sathasivam et al. 2020b, Section 2). The effect of an election algorithm (EA) on enhancing the learning process of an HNN to generate a solution for the RANKSAT logic representation is examined in Sathasivam et al. (2020b).

The MFTHNN-2SAT model employs an HNN to solve a 2SAT problem using a mean-field theory (MFT) algorithm (X4.5: heuristic optimization for SAT) (Sathasivam et al. 2020a, see Abstract). A 2SAT instance is a PL formula written in CNF with at most two literals per clause (X6.2: logic expressivity) (Sathasivam et al. 2020a, Section 5A). The integration of 2SAT into an HNN

using MFT comprises 11 steps (Sathasivam et al. 2020a, Section 5C). The input to this procedure is a 2SAT instance. In the final step, the procedure calculates the final energy for each neuron when the neurons are in a stable state. The output phase aims to find the best global solution, although the network may also reach a local minimum (Sathasivam et al. 2020a, Eq. 27). Satisfiability witness generation (X2.1) requires evidence that the procedure produces a truth assignment satisfying the input formula. The best global solution (Sathasivam et al. 2020a, see Abstract) alone does not explicitly establish this. Concepts from the Y dimension require explicit verification or soundness evidence. Calculating final energy alone does not establish these properties.

DHNN-3SATLOA combines a DHNN with the modified Lion optimization algorithm (LOA) for Boolean 3-satisfiability programming (X3.1.1.1: SAT solving). The Boolean 3SAT formula is written in CNF, with three literals per clause (X6.2: logic expressivity). The LOA improves the learning phase (X4.5: heuristic optimization for SAT), while the Wan Abdullah method determines synaptic weights. DHNN-3SATLOA computes the final energy and uses a tolerance-based comparison with the global minimum energy to classify the result as a local or global minimum (Mansor et al. 2020, Figure 1).

NeuroGlue uses a GNN to predict glue variables for SAT for CNF formulas (X3.1.1.1: SAT solving) (Han 2020, Sections 2–4). Its predictions periodically refocus CaDiCaL’s EVSIDS branching heuristic (X4.4: branching heuristic), while CaDiCaL performs the symbolic search (Han 2020, Section 5). No qualifying Y-dimension assurance mechanism is described in the paper.

The HNN-R2SATEA model incorporates the modified election algorithm into the learning phase of a DHNN to accelerate learning in optimizing random 2SAT logic representation (X3.1.1.1: SAT solving; X4.5: heuristic optimization for SAT) (Abubakar et al. 2020b, Section 3.2, Step 5; Algorithm 1). Random 2SAT is represented as a PL formula written in CNF with one or two literals per clause (X6.2: logic expressivity) (Abubakar et al. 2020b, Section 2.1, Eq. 8). HNN searches for an optimal solution to the cost function by satisfying the maximum number of clauses (Abubakar et al. 2020b, Section 2.1).

HornSAT logic programming represents a PL formula in the form of Horn clauses (Alzaeemi et al. 2020, Section II.2). HornSAT logic programming is implemented in HNN, with Horn clauses containing at most one positive literal (X6.2: logic expressivity) (Alzaeemi et al. 2020, Sections II.2 and II.4). The aim is to decide whether there exists an assignment of truth values to the variables in the clauses that makes the HornSAT logic program satisfiable (Alzaeemi et al. 2020, Section II.2; see Formula 4). The authors develop a procedure to search for a neuron state corresponding to a truth

Table 27. X–Y annotations and ANN architectures of NeSy frameworks, 2020.

ID	Reference	X and Y concept	ANN architecture
1	Zhang et al. (2020)	X2.1, X3.1.1.1 X4.5	GGC-NN
2	Kurin et al. (2020)	X3.1.1.1, X4.4	GNN
3	Riegel et al. (2020)	X1.2, X6.2 X8.2	LNN
4	Gauthier (2020)	X1.2	Tree-NN
5	Shi et al. (2020)	X5.1, X6.2	LINN
6	Vaezipoor et al. (2020)	X3.1.1.1, X3.1.2.2, X4.5	DNN
7	Sianipar et al. (2020)	X4.5, X6.2	HNN
8	Sathasivam et al. (2020c)	X3.1.1.1, X6.2	HNN
9	Azar et al. (2020)	X3.1.1.1, X6.2	MPNN
10	Sathasivam et al. (2020b)	X6.2	HNN
11	Sathasivam et al. (2020a)	X4.5, X6.2	HNN
12	Rasli et al. (2020)	X3.1.1.1, X4.5, X6.2	MHNN
13	Mansor et al. (2020)	X3.1.1.1, X4.5 X6.2	DHNN
14	Alzaeemi et al. (2020)	X3.1.1.1, X4.5 X6.2	HNN
15	Abubakar et al. (2020a)	X3.2, X6.2	HNN
16	Abubakar et al. (2020b)	X3.1.1.1, X4.5 X5.2	DHNN
17	Han (2020)	X3.1.1.1, X4.4	GNN

assignment that satisfies the input HornSAT formula (X3.1.1.1: SAT solving) by constructing synaptic weights and updating neuron states (X4.5: heuristic optimization for SAT) (Alzaeemi et al. 2020, Section II.4, pp. 1874–1875).

Random Maximum kSatisfiability (MAXRkSAT) is a class of PL formulas in CNF (X6.2: logic expressivity) (Abubakar et al. 2020a, Section 2). HNN-MAXRkSAT integrates MAXRkSAT into an HNN (X3.2: MaxSAT optimization). HNN uses the Wan Abdullahi method to compute synaptic weights (Abubakar et al. 2020a, Sections 4, Eq. 15).

The MHNN-2SAT model combines HNN and estimation of distribution algorithm (EDA) for 2SAT logic programming (X3.1.1.1: SAT solving), with $k = 2$ literals per clause (X6.2: logic expressivity). HNN uses energy minimization. EDA is implemented during the retrieval phase, using Univariate Marginal

Gaussian distribution to alter the final neuron state (X4.5: heuristic optimization for SAT) (Rasli et al. 2020, pp. 2–3). The procedure translates all clauses into Boolean algebra, randomly initializes the HNN input, computes the HNN output, calculates the mean, variance and joint probability distribution function, and normalizes the latter. It then obtains a new neuron state, updates the best solution using roulette-wheel selection, retains the new output, and repeats the process until $f_{2SATES} = NC$, where NC is the total number of 2SAT clauses (Rasli et al. 2020, p. 3, Steps 1–8).

The paper describes LNN and its reasoning capability restricted to PL (Riegel et al. 2020, Section 3). In LNN, neurons correspond to propositions and logical operations (X6.2: logic expressivity; X8.2: interpretability). Bidirectional inference performed by LNN derives truth value bounds for a connective’s inputs from bounds on its output and other inputs (X1.2: logical consequence) (Riegel et al. 2020, Section 4.1). The Upward-Downward algorithm propagates these bounds through formula syntax trees (Riegel et al. 2020, Section 4.3).

The Tree-NN is implemented within the HOL proof assistant (Gauthier 2020, see Abstract). The author trains the Tree-NN to determine whether a PL formula in the form $A \implies B$ is true under every truth assignment, equivalently, whether B is a logical consequence of A (X1.2: logical consequence) (Gauthier 2020, Section 4).

LINN is a dynamic ANN architecture that combines logic reasoning with deep learning techniques Shi et al. (2020). In the LINN, each propositional letter and each logical constant $\top$, $\perp$ is represented as a vector embedding, while logical connectives $\wedge$, $\vee$, $\neg$ are implemented as learnable neural modules (X6.2: logic expressivity) (Shi et al. 2020, Section 3). Given a PL formula as input, the LINN dynamically builds a graph. The LINN performs reasoning through this graph and predicts truth values of PL expressions written in DNF using observed variables (X5.1: logical generalization) (Shi et al. 2020, see Abstract).

In (Vaezipoor et al. 2020, Sections 1–2), the authors train a DNN to guide the Glucose SAT solver Audemard and Simon (2018) to solve the SAT problem in PL (X3.1.1.1: SAT solving). A policy gradient algorithm optimizes a stochastic policy that outputs an integer literal block distance (LBD) threshold (Vaezipoor et al. 2020, p. 2, Section 2, see Action paragraph). Learned clauses with LBD values above this threshold are deleted (X3.1.2.2: RL-guided clause selection). Training rewards fewer clause accesses during unit propagation (X4.5: heuristic optimization for SAT) (Vaezipoor et al. 2020, p. 2, Section 2, see Reward paragraph).

The NeurNet-SP algorithm combines survey propagation with an FNN to construct approximate assignments for MAX-E-3-SAT (Marino 2021,

Section 4.2, Algorithm 2). The method minimizes the number of unsatisfied clauses into an instance of MAX-E-3-SAT (X3.2: MaxSAT optimization). Satisfiability witness generation is not assigned to the publication Marino (2021).

In Tran and d'Avila Garcez (2021), authors prove that any PL formula through strict DNF can be mapped into RBMs (X6.2: logic expressivity) (Tran and d'Avila Garcez 2021, Section 3). The translation has a proof of equivalence (X7: soundness guarantee; Y3: soundness assessment)(Tran and d'Avila Garcez 2021, Theorem 1). In experiments, Gibbs sampling finds a truth assignment that satisfies at least one conjunction of literals and therefore satisfies the PL formula written in DNF (X2.1: satisfiability witness generation; X3.1.1.1: SAT solving) (Tran and d'Avila Garcez 2021, Section 6).

RAN- k -SAT is a class of PL formulas written in CNF, with clauses containing one, two, or three literals for $k \leq 3$ (X6.2: logic expressivity) (Abubakar et al. 2021, p. 202, Eqs. (1)–(3)) (Abubakar and Danrimi 2021, p. 662, Eqs. (1)–(4)). RAN- k -SAT is encoded in an HNN. The election algorithm (EA) is applied in the training phase of the HNN to search for optimal representation solutions and to improve or follow preferred assignments (X3.1.1.1: SAT solving; X2.1: satisfiability witness generation; X4.5: heuristic optimization for SAT) (Abubakar et al. 2021, see Conclusion) (Abubakar and Danrimi 2021, p. 668).

RAN3SAT is represented as PL formulas in CNF with clauses containing at most three literals and formulated as P_{RAN3SAT} (X6.2: logic expressivity) (Bazuhair et al. 2021, p. 4, Eqs. (1)–(3)). The election algorithm (EA) is implemented in an HNN and learns all the logical combinations of RAN3SAT to find the optimal assignment for P_{RAN3SAT} leading to $E_{P_{\text{RAN3SAT}}} = 0$ (X3.1.1.1: SAT solving; X2.1: satisfiability witness generation; X4.5: heuristic optimization for SAT) (Bazuhair et al. 2021, pp. 12–13).

As in Bazuhair et al. (2021), Karim et al. (2021), RAN3SAT represents PL formulas written in CNF, with clauses containing one, two, or three literals in different logical combinations (X6.2: logic expressivity) (Karim et al. 2021, p. 50833, Eqs. (1)–(7)). This structure is integrated with a DHNN through the cost function and Lyapunov energy function (Karim et al. 2021, pp. 50833–50834, Eqs. (8)–(16)). The exhaustive search (ES) algorithm facilitates the training phase to check clause satisfaction, and a zero value of the cost function indicates that the logical rule is satisfied (X3.1.1.1: SAT solving).

The study Mansor et al. (2021) develops a DHNN-3SATGWO model for the SAT problem in PL, with formulas expressed in CNF (X3.1.1.1: SAT solving). The model uses bipolar Grey Wolf Optimization to minimize the cost function during the learning phase of a DHNN (Mansor et al. 2021, pp. 4–6, Section 3). The learning process is completed when the model obtains satisfied states that lead to $E_{\kappa_{3SAT}} = 0$ (Mansor et al. 2021, p. 3, Section 2, Eq. 5). The

Table 28. X–Y annotations and ANN architectures of NeSy frameworks, 2021.

ID	Reference	X and Y concept	ANN architecture
1	Tran and d'Avila Garcez (2021)	X2.1, X3.1.1.1, X6.2, X7, Y3	RBM
2	Marino (2021)	X3.2	FNN
3	Wang et al. (2021)	X1.1, X6.2 X3.1.1.1, X4.5	LSTM RNN
4	Mansor and Sathasivam (2021) , Mansor et al. (2021)	X3.1.1.1	DHNN
5	Bazuhair et al. (2021)	X2.1, X3.1.1.1, X4.5, X6.2	HNN
6	Karim et al. (2021)	X3.1.1.1, X6.2	DHNN
7	Abubakar et al. (2021) Abubakar and Danrimi (2021)	X6.2, X3.1.1.1, X2.1, X4.5	HNN

paper does not mention that the DHNN-3SATGWO model supports formal verification, interactive explanations, or soundness assessments.

The study [Mansor and Sathasivam \(2021\)](#) evaluates a DHNN-3SAT model for the SAT problem in PL in which formulas are written in CNF (X3.1.1.1: SAT solving) ([Mansor and Sathasivam 2021](#), p. 965, Section 2). The DHNN uses the Wan Abdullah method for synaptic-weights calculation and seeks satisfied states with $E_{\omega_{3SAT}} = 0$ ([Mansor and Sathasivam 2021](#), p. 966, Eqs. 2.5). Final neuron states are assessed through global and local minimum energy ([Mansor and Sathasivam 2021](#), p. 971, Figure 1). No UNSAT refutation or formal assurance is reported.

A heuristic algorithm based on an LSTM RNN is proposed in ([Wang et al. 2021](#), see Abstract) for SAT solving problems in PL with formulas expressed in CNF (X3.1.1.1: SAT solving; X4.5: heuristic optimization for SAT; X6.2: logic expressivity). The trained end-to-end SAT solver outputs 0 or 1 as a representation of SAT candidates. However, the model is also used together with an LSTM-DPLL algorithm, which has an explicit UNSAT return ([Wang et al. 2021](#), p. 143). Therefore, the combined approach supports satisfiability decision (X1.1), whereas X2.2 is not assigned.

The HNN-2SAT model uses ACO during its learning phase to optimize the search for a solution to 2SAT, that is, SAT for CNF formulas in which each clause contains two literals (X3.1.1.1: sat solving; X4.5: heuristic optimization for sat) ([Kho et al. 2022](#), p. 40, Section 2.2, and pp. 42–45). The authors do not integrate a standard symbolic 2SAT solver with the HNN. Instead, 2SAT logic is implemented in the HNN by assigning one neuron to each PL variable and mapping the clause cost function to the HNN energy

function (X2.1: satisfiability witness generation) (Kho et al. 2022, pp. 40–42) (Kho et al. 2022, pp. 43–45). The method does not produce an UNSAT refutation or a formal proof, and failure to find a satisfying assignment does not establish UNSAT. No assurance mechanism corresponding to the Y dimension was identified.

An RNN with a formula query mechanism, named QuerySAT, is a recurrent neural SAT solver for solving SAT problem in PL (Ozolins et al. (2022) (X3.1.1.1: SAT solving). PL formulas are represented in CNF (X6.2: logic expressivity). During the reasoning process, its query mechanism generates a candidate solution (query). This query is evaluated using an unsupervised loss function that measures how well it satisfies the given SAT instance, and the evaluation result is fed back to the RNN for further interpretation (X3.1.2.3: query-aware formula generation).

Table 29. X–Y annotations and ANN architectures of NeSy frameworks, 2022.

ID	Reference	X and Y concept	ANN architecture
1	Zhang et al. (2022b)	see Zhang et al. (2023) in Table 30	BERT
2	Fan and Zhang (2022)	X6.1, X6.2	Noan
3	Kho et al. (2022)	X2.1, X3.1.1.1, X4.5	HNN
4	Ozolins et al. (2022)	X3.1.1.1, X3.1.2.3, X6.2	RNN
5	Zhang et al. (2022a)	X1.1, X3.1.1.1 X5.1, X2.1	TenSAT
6	Zamri et al. (2022a), Zamri et al. (2022b), Zamri et al. (2022c)	X2.1, X6.2 X4.5,X3.1.1.1	DHNN
7	Someetheram et al. (2022)	X6.2, X3.2 X4.5, X2.1	DHNN
8	Li and Si (2022)	X2.1, X3.1.1.1, X6.2	GNN
9	Karim et al. (2022)	X2.1, X6.2, X3.1.1.1	DHNN
10	Gao et al. (2022)	see Gao et al. (2024) in Table 31	
11	Guo et al. (2022)	see Guo et al. (2024) in Table 31	
12	Chang et al. (2022)	X1.1, X3.1.1.1, X6.2	GNN
13	Alway et al. (2022)	X2.1, X6.2	DHNN
14	Abubakar (2022)	X3.2	HNN
15	Freivalds and Kozlovics (2022)	X3.1.1.1	GNN

DiffusionSat combines a denoising diffusion algorithm with a recurrent GNN to generate solutions for SAT instances expressed in CNF (X3.1.1.1:

SAT solving) Freivalds and Kozlovics (2022). For randomly generated CNF with three literals, training instances contain 5–100 variables, while accuracy tests use instances with up to 405 variables (Freivalds and Kozlovics 2022, Appendix A).

Neural logic analogy learning (Noan) is a neural architecture presented in Fan and Zhang (2022) that was developed to solve letter-string analogy problems. In this approach, each analogy problem is converted into PL expressions, and candidate answers are ranked using the predicted true or false scores of those expressions (Fan and Zhang 2022, pp. 2–3, Sections 2.2 and 2.4). Each alphabetic character $x \in A, \dots, Z$ is represented by a PL variable v_x , while the characters occurring in a string are connected by conjunction. Noan does not introduce a separate PL variable for each string position (Fan and Zhang 2022, p. 2, Section 2.2). The Noan ANN learns PL variables as vector embeddings and each PL connective ($\vee, \wedge, \neg$) as a neural module to evaluate the resulting PL expressions (X6.1: algorithmic alignment). The neural modules are trained using soft logical regularizers corresponding to selected laws of PL. (Fan and Zhang 2022, pp. 2–3, Sections 2.2–2.3) (X6.2: logic expressivity). Noan builds a computational graph that integrates the ANN with logical reasoning to capture the structural information of the analogy expressions (Fan and Zhang 2022, see Abstract). Noan predicts a score in $[0, 1]$ for each expression by comparing its vector representation with a fixed True vector $\mathbf{T}$ and uses this score to rank a manually constructed set of candidate answers (Fan and Zhang 2022, pp. 2–3, Sections 2.2 and 2.4). This learned scoring procedure is neither a satisfiability decision (X1.1: satisfiability decision) nor a logical consequence (X1.2: logical consequence), and it is not reasoning performed by a symbolic PL engine. The paper provides no assurance mechanism corresponding to Y1: verification, Y2: interactive explanation, or Y3: soundness assessment.

In Zhang et al. (2022a), the authors design the TenSAT SAT solver that improves performance with numerous PL variables (Zhang et al. 2022a, see Abstract). TenSAT addresses SAT for CNF formulas by predicting satisfiability (Zhang et al. 2022a, p. 204) (X1.1: satisfiability decision; X3.1.1.1: SAT solving; X2.1: satisfiability witness generation). Tests on formulas with more propositional variables evaluate generalization (Zhang et al. 2022a, pp. 205–209) (X5.1: logical generalization).

An r2SAT logic structure that represents a class of PL formulas written in CNF is embedded as the logical rule in a DHNN Zamri et al. (2022a). Each PL literal in the r2SAT logic structure is implemented as a neuron in the DHNN (X6.2: logic expressivity). Metaheuristic is used to minimize logical inconsistency when searching for satisfying assignments (X4.5: heuristic optimization for SAT; X3.1.1.1: SAT solving). Specifically, the

training algorithms minimize the r2SAT cost function ($E_{\Gamma_{\text{r2SAT}}}$) (Zamri et al. 2022a, Formula 13) to search for satisfying assignments of the r2SAT formula (Γ_{r2SAT}). The resulting optimal DHNN neuron states encode truth assignments that satisfy every clause and therefore satisfy the PL formula written in CNF (Zamri et al. 2022a, Section 7.2, p. 14) (X2.1: satisfiability witness generation).

Closely related to the previous work Zamri et al. (2022a), the study Zamri et al. (2022b) embeds the r2SAT formula, written in CNF with clauses containing one and two literals, as a logical rule in a DHNN. Bipolar neuron states represent Boolean interpretations (X6.2: logic expressivity). A genetic algorithm minimizes the cost function to search for satisfying assignments (X2.1: satisfiability witness generation; X4.5: heuristic optimization for SAT; X3.1.1.1: SAT solving). The paper provides no SAT or UNSAT decision, formal SAT proof, certificate, or proof-validation mechanism. Therefore, it receives an empty Y annotation set.

In Zamri et al. (2022c), the authors compare the DHNN-r2SAT and DHNN-3SAT models (Zamri et al. 2022c, p. 5, see Experimental settings). The r2SAT and 3SAT logical rules are embedded in a DHNN, where bipolar neuron states represent Boolean interpretations (X6.2: logic expressivity). Exhaustive search in both models minimizes logical inconsistency to find satisfying assignments, while DHNN dynamics retrieve global minimum states (X2.1: satisfiability witness generation; X4.5: heuristic optimization for SAT; X3.1.1.1: SAT solving). The framework provides no UNSAT refutation, resolution or entailment proof, certificate, proof validation, or sound-and-complete SAT decision procedure. Therefore, its Y annotation set is empty. Overall, DHNN-3SAT model finds satisfying truth assignments more often than DHNN-r2SAT (Zamri et al. 2022c, p. 9, see Table 5).

In Someetheram et al. (2022), Random Maximum 2 Satisfiability (RAN2SAT), a class of PL formulas in CNF with two literal clauses (Someetheram et al. 2022, see Formula 2), is embedded in a DHNN, whose bipolar states encode Boolean assignments (X6.2: logic expressivity). The improved election algorithm heuristically maximizes satisfied clauses (X3.2: MaxSAT optimization; X4.5: heuristic optimization for SAT). An assignment that satisfies all clauses is a witness (X2.1: satisfiability witness generation). However, it does not provide a complete SAT decision (X1.1: satisfiability decision), UNSAT refutation (X2.2: UNSAT refutation), certificate support (Y1.1: certificate support), or proof validation (Y1.2: proof validation). Thus, no Y concept is assigned.

NSNet Li and Si (2022) is a NeSy framework that performs SAT solving (X3.1.1.1: SAT solving) with satisfiability witness generation (X2.1: satisfiability witness generation) for PL formulas written in CNF (X6.2: logic

expressivity). The core neural component is a GNN that parameterizes the Belief Propagation algorithm in the latent space and performs marginal inference among all satisfying solutions of the CNF formula (Li and Si 2022, Section 3.3, Equation 8). A truth assignment can be generated by rounding and executing a stochastic local search (Li and Si 2022, see Abstract).

In Karim et al. (2022), the authors integrate a class of PL formulas, namely RANkSAT, containing clauses with two or three literals, into a DHNN (X6.2: logic expressivity). Thus, a DHNN-RANkSAT model directly supports both SAT solving (X3.1.1.1: SAT solving) and satisfiability witness generation (X2.1: satisfiability witness generation). A satisfiability witness is generated when the DHNN reaches a global minimum energy. However, the DHNN-RANkSAT model Karim et al. (2022) does not provide UNSAT refutation, proof generation, proof validation, or a general SAT or UNSAT decision procedure.

The graph attention network model (GAT-SAT) predicts the satisfiability of stochastic SAT instances represented as PL formulas in CNF (X3.1.1.1: SAT solving; X6.2: logic expressivity) (Chang et al. 2022, p. 84, see Figure 1). Each SAT instance is encoded as an undirected graph represented by an adjacency matrix (Chang et al. 2022, Section 3). The GAT-SAT model takes as input only the adjacency matrix, which may represent any number of literals and clauses (Chang et al. 2022, Section 3.2). A score above (0.5) results in classification as SAT; otherwise, the PL formula is classified as UNSAT (X1.1: satisfiability decision).

The Major 2 Satisfiability (MAJ2SAT) is a restricted class of PL formulas in CNF containing clauses with two or three literals (Alway et al. 2022, p. 926). This MAJ2SAT is implemented into DHNN, resulting in the DHNN-MAJ2SAT model. The DHNN-MAJ2SAT flowchart consists of a training phase and testing phase (Alway et al. 2022, see Figure 2). DHNN-MAJ2SAT takes the user-defined ratio ($\alpha = \frac{m}{m+n}$) as input. It determines $\varphi_{\text{MAJ2SAT}} = \bigwedge_i^m A_i^{(2)} \wedge \bigwedge_i^n A_i^{(3)}$ (X6.2: logic expressivity). Training checks clause satisfaction until $E_{\varphi_{\text{MAJ2SAT}}} = 0$, and stores synaptic weights (X2.1: satisfiability witness generation). Testing relaxes the DHNN and classifies final energy as a global or local minimum.

In Abubakar (2022), the author explicitly maps RANMAXkSAT PL formulas written in CNF, with at most three literals per clause, to an HNN. The energy function of the HNN is used as a dynamic minimization mechanism (Abubakar 2022, see Abstract). Therefore, the framework supports the optimization of RANMAXkSAT logical formulas. This functionality is mapped to MaxSAT optimization (X3.2) in the X-Y-Z taxonomy. It should not be classified as satisfiability decision (X1.1) or satisfiability witness generation (X2.1) because the optimization does not require every clause to be satisfied.

No UNSAT refutation, entailment, resolution, proof, or other assurance mechanism is provided.

In Tran and Garcez (2023), the authors introduce an LBM that can represent any PL formula (Tran and Garcez 2023, see Abstract). The authors also prove the equivalence between PL and RBM (Tran and Garcez 2023, Introduction). Specifically, a PL formula can be encoded in an LBM by translating it into a DNF. The authors demonstrate that the resulting LBM can determine that a PL formula is SAT (X1.1: satisfiability decision); see Table 30) without the need for training data. However, the LBM cannot deterministically prove that a PL formula is UNSAT. Additionally, the framework presented in Tran and Garcez (2023) does not perform theorem proving in PL.

In Liu et al. (2023), authors describe two GNN models, GMS-N and GMS-E, that encode PL formulas in CNF as clause variable-incidence graphs (Liu et al. 2023, Figure 1). These models are trained to predict the solution of the MaxSAT problem (X3.2: MaxSAT optimization). Evaluation of these models on larger instances demonstrates logical generalization (X5.1).

Table 30. X–Y annotations and ANN architectures of NeSy frameworks, 2023.

ID	Reference	X and Y concept	ANN architecture
1	Liu et al. (2023)	X3.2, X5.1	GNN (GMS-N, GMS-E)
2	Tran and Garcez (2023)	X1.1	LBM
3	Shi et al. (2023)	X5.3, X1.1	GNN
4	Roslan et al. (2023)	X3.1.1.1 X2.1, X4.5, X3.1.1.1, X6.2	DHNN
5	Moriyama et al. (2023)	X4.2, X5.3	GNN
6	Li et al. (2023)	X2.1, X3.1.1.1 X4.5	GNN
7	Hu et al. (2023)	X3.2, X4.3 X4.5	GNN
8	Azizan and Sathasivam (2023)	X3.1.1.1, X4.5	HNN
9	Azizan et al. (2023a)	X3.1.1.1, X4.5	HNN
10	Azizan et al. (2023b) Ali et al. (2023)	X6.2, X2.1 X3.1.1.1, X4.5	HNN
11	Abdeen et al. (2023)	X6.2, X2.1 X2.1, X6.2 X3.1.1.1	DHNN
12	Zhang et al. (2023)	X1.2, X6.2	BERT
13	Tönshoff et al. (2023)	X3.1.1.1, X3.2, X4.5	GNN

The ANYCSP method uses a recurrent GNN to process general CSP instances represented as constraint value graphs (Tönshoff et al. 2023, p. 4281, Figure 1; p. 4282, Section 4.1). For SAT instances expressed as PL formulas in CNF, it predicts probabilities over variable values and samples candidate truth assignments during global search (X3.1.1.1: SAT solving) (Tönshoff et al. 2023, Sections 4.2 and 5). For MaxSAT, ANYCSP searches for assignments that maximize the number of satisfied clauses (X3.2: MaxSAT optimization) (Tönshoff et al. 2023, p. 4286, Table 4). The GNN is trained using REINFORCE with stochastic policy gradient ascent to serve as a global search heuristic for CSPs, including SAT (X4.5: heuristic optimization for SAT) (Tönshoff et al. 2023, Section 4.2, pp. 4283–4284). The reported evidence does not establish an assurance mechanism eligible for Y annotation.

SimpleLogic uses definite clauses of the form $A_1 \wedge A_2 \wedge \dots \wedge A_n \rightarrow B$ in reasoning tasks in PL (X6.2: logic expressivity) (Zhang et al. 2022b, p. 3366, Section 2.1). The reasoning task is to determine whether a query atom can be proved from the facts and rules. More formally, prove B when each of $A_1, \dots, A_n$ can be proved from the propositional theory (X1.2: logical consequence). An example of a SimpleLogic reasoning problem sampled using the rule-priority and label-priority methods is illustrated in (Zhang et al. 2023, p. 3368, Figure 3). BERT performs well on examples drawn from its training distribution but fails to generalize to other distributions over the same reasoning problem space (X5.1: logical generalization).

SATformer explicitly addresses satisfiability prediction for PL formulas in CNF using a GNN. It is trained using a multi-task learning approach (Shi et al. 2023, Abstract) and incorporates clause-level UNSAT core learning (X5.3: UNSAT-core prediction) to support formula classification. However, its neural outputs are approximate and should not be treated as proofs or certified UNSAT-cores (X1.1: satisfiability decision; X3.1.1.1: SAT solving). A predicted UNSAT-core is neither a resolution refutation nor an independently verified UNSAT certificate. A Y concept is not assigned to the SATformer framework because no certificate, external verification, proof validation, or soundness assessment is reported Shi et al. (2023).

Major 3 Satisfiability (MAJ3SAT) logic structure is embedded in a DHNN. MAJ3SAT is a restricted class of PL formulas in CNF, where clauses with three literals are predominant and combined with clauses containing one or two literals (X6.2: logic expressivity) (Roslan et al. 2023, pp. 22451–22453, Sections 3–4.1). During learning, the election algorithm searches candidate neuron states to maximize the fitness of each neuron and minimize the MAJ3SAT cost function (X4.5: heuristic optimization for SAT). A fully satisfied interpretation has zero cost (X2.1: satisfiability witness generation; X3.1.1.1:

SAT solving). No assurance mechanism within the scope of the Y dimension was identified in the paper.

The GNN-MUS model is a trainable GNN that extracts minimal unsatisfiable subsets of PL clauses. These subsets serve as explanations for the unsatisfiability (Moriyama et al. 2023, Abstract) (X5.3: UNSAT-core prediction; X4.2: clause selection quality). Rather than providing a certified UNSAT refutation, it supplies clause predictions, and no Y concept is assigned (Moriyama et al. (2023)).

A PL formula in CNF is converted into a directed acyclic GNN representation Li et al. (2023). A trained GNN predicts an initial assignment for a local-search SAT solver (X4.5: heuristic optimization for SAT). DeepSAT produces satisfying assignments for the SAT instance from the trained conditional generative model using a simple solution sampling scheme (X2.1: satisfiability witness generation; X3.1.1.1: SAT solving) (Li et al. 2023, see Introduction).

A framework presented in Hu et al. (2023) formulates the problem of fixing privilege escalations in cloud access control as a MaxSAT problem. It then uses a GNN to improve the scalability of MaxSAT solving (X3.2: MaxSAT optimization; X4.5: heuristic optimization for SAT). The GNN predicts a subset of candidate repair operations, thereby restricting the repair space that the MaxSAT solver searches. The MaxSAT solver then computes the final repair (X4.3: search-space pruning). However, the paper does not report a proof certificate or assurance mechanism corresponding to the Y dimension, so no Y concept is assigned to the framework.

In Azizan and Sathasivam (2023), the authors integrate a PL formula in CNF with clauses containing exactly three literals and an HNN. They use randomized alpha-cut fuzzy logic to modify unsatisfied clauses while searching for a global minimum neuron state. This approach directly supports heuristic SAT solving in PL (X3.1.1.1: SAT solving; X4.5: Heuristic optimization for SAT). However, the paper does not establish a SAT or UNSAT decision, an UNSAT refutation, or proof assurance. Accordingly, the paper does not clearly report a complete truth assignment verified against the entire formula.

In Azizan et al. (2023a), the authors work on finding assignments that make an entire PL formula true. The 3SAT logical structure, representing a class of PL formulas written in CNF with exactly three literals in each clause, is embedded in an HNN (X6.2: logic expressivity), while a genetic algorithm optimizes the candidate neuron states, returning them in bipolar form as candidate satisfying assignments. This capability is assigned the survey annotations satisfiability witness generation (X2.1), SAT solving (X3.1.1.1), and heuristic optimization for SAT (X4.5). The framework does not decide

UNSAT or produce an UNSAT refutation. Therefore, X1.1 and X2.2 are not assigned. MAX-3SAT is mentioned only as future work. No Y concept is supported.

In [Azizan et al. \(2023b\)](#), Boolean satisfiability is defined over PL formulas written in CNF, with emphasis on 3SAT formulas containing three literals per clause. The HNN-3SATFuzzy model encodes the 3SAT in an HNN (X6.2: logic expressivity). Heuristic optimization techniques based on fuzzy logic expand the search over candidate neuron states, convert candidate states into crisp values, and adjust neuron states associated with unsatisfied clauses (X4.5: heuristic optimization for SAT). The resulting interpretation supports the survey annotations X2.1: satisfiability witness generation and X3.1.1.1: SAT solving. The paper reports neither a complete SAT or UNSAT decision nor an UNSAT refutation. No Y concept is supported.

In [Ali et al. \(2023\)](#), exact Boolean (k)-satisfiability is formulated over PL formulas written in CNF, with the requirement that exactly one literal be true in every clause. Each bipolar neuron string represents a truth assignment, and the artificial dragonfly algorithm searches these assignments. A satisfying assignment is stored in the HNN memory component (X2.1: satisfiability witness generation; X3.1.1.1: SAT solving; X4.5: heuristic optimization for SAT; X6.2: logic expressivity). However, the framework does not establish a complete SAT or UNSAT decision nor produce an UNSAT refutation. This implies that concepts X1.1 and X2.2 are not assigned. No assurance mechanism corresponding to the Y dimension is reported.

In [Abdeen et al. \(2023\)](#), S-Type Random 2SAT (S2SAT) logic represents PL formulas in CNF with clauses containing one or two literals. The S2SAT logic is embedded in a DHNN (X6.2: logic expressivity). The framework minimizes a logic-derived cost function, where a zero-cost state represents a consistent interpretation that satisfies all clauses (X3.1.1.1: SAT solving). The final neuron states encode the corresponding truth assignment (X2.1: satisfiability witness generation). However, the paper does not provide a complete SAT and UNSAT decision procedure nor an UNSAT refutation. There is no reported certificate, proof-validation procedure, or other assurance mechanism in the Y dimension.

The NeuroBack approach enhances a CDCL SAT solver with GNN guidance [Wang et al. \(2024a\)](#). NeuroBack converts a PL formula written in CNF into a graph representation and uses a GNN to predict variable phases that guide the CDCL SAT solver. The CDCL SAT solver, not the GNN, performs the satisfiability search and returns SAT or UNSAT result (X1.1: satisfiability decision; X3.1.1.1: SAT solving; X4.5: heuristic optimization for SAT). No Y assurance mechanism is reported ([Wang et al. 2024a](#), p. 4, Section 4.1).

Table 31. X–Y annotations and ANN architectures of NeSy frameworks, 2024.

ID	Reference	X and Y concept	ANN architecture
1	Wang et al. (2024a)	X1.1, X3.1.1.1, X4.5	GNN
2	Ghanem (2024)	X2.2, X3.1.1.2, Y1.1, Y1.1.1, Y1.4.1	GNN
3	Xie et al. (2024)	X4.5, X2.1 X3.1.1.1	DHNN
4	Rusdi et al. (2024)	X2.1, X3.1.1.1 X4.5, X6.2	DHNN
5	Roslan et al. (2024)	X2.1, X3.1.1.1 X6.2, X4.5	DHNN
6	Manoharam et al. (2024)	X2.1 X3.1.1.1, X4.5 X6.2	DHNN
7	Jiang et al. (2024)	X6.2, X3.1.1.1	DHNN
8	Guo et al. (2024)	X3.1.1.1, X4.5 X6.2	DHNN
9	Gao et al. (2024)	X2.1, X4.5 X6.2, X3.1.1.1	DHNN
10	Feng and Sathasivam (2024)	X6.2, X3.1.1.1	DHNN
11	Feng et al. (2024)	X2.1, X4.3, X4.5 X6.2, X3.1.1.1	DHNN
12	Chen et al. (2024)	X4.3, X4.5 X6.2	DHNN
13	Chang et al. (2024)	X1.1, X3.1.1.1 X5.1, X6.2	GRU-NN
14	Farooque et al. (2024)	X1.1, X4.4, X4.5 X6.2, X3.1.1.1	GAT
15	Hüla et al. (2024)	X1.1, X2.1, X3.1.1.1, X3.2, X4.3, X4.5 X5.1, X6.2	GNN NeuroSAT
16	Zhang et al. (2024b)	X1.1, X3.1.1.1, X3.1.2.4	GNN GraSS
17	Zheng et al. (2024)	X3.1.2.3, X3.1.1.1, X1.1, X6.2	GNN
18	Lymperopoulos and Liu (2024)	X4.3, X1.1 X3.1.1.1	GNN

The ANF-based graph representation encodes XOR and AND operations in cryptographic SAT instances (X6.2: logic expressivity). For each key bit, two extended instances fix it to 0 or 1 (X3.1.2.3: query-aware formula generation). CryptoANFNet, a supervised GNN classifier, produces satisfiability scores (X1.1: satisfiability decision; X3.1.1.1: SAT solving). Repeating the higher-score selection yields a candidate key assignment. The authors report no verification, interactive explanation, or soundness assessment mechanism [Zheng et al. \(2024\)](#).

NeuRes, a NeSy generative model, is presented in [Ghanem \(2024\)](#). Unlike NeuroBack, which uses GNN guidance for CDCL SAT solving ([Wang et al. 2024a](#), p. 4, Section 4.1), NeuRes generates verifiable resolution proofs for unsatisfiable PL formulas expressed in CNF (X2.2: UNSAT refutation; X3.1.1.2: resolution; Y1.1: certificate support; Y1.1.1: certificate format; Y1.4.1: UNSAT certification). Its architecture employs a GNN to embed a PL formula written in CNF ([Ghanem 2024](#), p. 10, Section 3.3). A message-passing GNN is used to obtain clause embeddings ([Ghanem 2024](#), p. 12, Section 4.2). The NeuRes model is trained via teacher forcing on expert proofs

(Ghanem 2024, p. 21, Section 5.2). It is designed to produce easily checkable UNSAT certificates (Y1.1: certificate support) in the form of resolution proofs (Ghanem 2024, see Abstract).

The DHNN-3SAT model Xie et al. (2024) integrates a DHNN with a genetic algorithm optimized K-modes clustering for Boolean SAT problems (X4.5: heuristic optimization for SAT) involving CNF formulas in which each clause has at most three literals (X2.1: satisfiability witness generation; X3.1.1.1: SAT solving). However, it does not establish general SAT/UNSAT decision capability (X1.1) because search failure does not imply UNSAT. It provides no UNSAT refutation (X2.2), certificate support (Y1.1), external verification (Y1.1.2), proof validation (Y1.2), or refutation certification (Y1.4). Its empirical results provide no general soundness guarantee and assessment (X7; Y3).

Negative Based Higher Order Systematic Satisfiability (NR3SAT) represents a restricted class of CNF in which every three-literal clause contains at least one negative literal (X6.2: logic expressivity) (Rusdi et al. 2024, Formula 3). The NR3SAT is embedded in a DHNN Rusdi et al. (2024). The DHNN states encode truth assignments, and a cost of zero indicates that every clause is satisfied (X2.1: satisfiability witness generation; X3.1.1.1: SAT solving) (Rusdi et al. 2024, pp. 29826–29829, Sections 3–4, Eqs. (1)–(13), Algorithm 1). The hybrid black hole algorithm (HBHA) optimizes the retrieval phase of DHNN (X4.5: heuristic optimization for SAT) (Rusdi et al. 2024, p. 29837, Algorithm 2). No Y concept is assigned because no assurance mechanism is reported.

CRAN2SAT is a restricted class of PL formula in CNF in which a set of literals can be either positive or negative (X6.2: logic expressivity) (Roslan et al. 2024, p.3916, Section 3). The CRAN2SAT formula is encoded in the DHNN cost function and synaptic weights. During retrieval, the Smish function updates the states of the DHNN neurons and yields bipolar outputs (X3.1.1.1: SAT solving; X4.5: heuristic optimization for SAT) (Roslan et al. 2024, p.3917, Section 3). The resulting bipolar values collectively represent a candidate CRAN2SAT truth assignment (X2.1: satisfiability witness generation). The bipolar values (1) and (-1) denote true and false, respectively. Thus, ($P_{\text{CRAN2SAT}} = 1$) when every clause is satisfied and ($P_{\text{CRAN2SAT}} = -1$) otherwise (Roslan et al. 2024, p. 3917, Section 3). The method produces neither UNSAT refutations nor proofs. No Y concept is assigned to the CRAN2SAT.

In the DHNN-MR1,3SAT approach, the MR1,3SAT logic defines a restricted class of randomly generated PL formulas in CNF (X6.2: logic expressivity) (Manoharam et al. 2024, Formulas 6 and 7). The DHNN embeds MR1,3SAT logic by representing each propositional variable as a Hopfield neuron.

During retrieval, the DHNN relaxes a random bipolar state toward an energy minimum. The final state is treated as a satisfying assignment only if its energy is within the prescribed tolerance of the predefined global minimum (X2.1: satisfiability witness generation; X3.1.1.1: SAT solving; X4.5: heuristic optimization for SAT) (Manoharam et al. 2024, p. 12094, Section 1; pp. 12101–12102; Figure 2; Algorithm 1). There is no description of an explicit certificate support, external verification, or refutation assurance mechanism. Therefore, no Y concept is assigned to the DHNN-MR1,3SAT.

In (Jiang et al. 2024, p. 3320, Section 2, Eqs. (1)–(2)), PL formulas written in CNF, with two or three literals per clause, are embedded in a DHNN (X6.2: logic expressivity). A random formula from this class is generated, and its clauses are checked under a candidate interpretation. When all clauses are satisfied by the interpretation, synaptic weights are computed and stored in the DHNN. The network retrieves final neuron states through local-field computation, and the energy of each state determines whether it is classified as a global- or local-minimum solution (Jiang et al. 2024, p. 3320, Figure 1). The resulting DHNN aims to retrieve global-minimum neuron-state vectors that encode truth assignments satisfying the corresponding PL formula (X2.1: satisfiability witness generation; X3.1.1.1: SAT solving) (Jiang et al. 2024, pp. 3321–3323, Section 3, Eqs. (7)–(14)). The paper does not report a SAT or UNSAT decision or an UNSAT refutation. It also reports no assurance mechanism corresponding to the Y dimension; therefore, no Y concept is assigned to the framework.

The framework embeds PL formulas written in CNF into a dynamic evolution variant of the HNN (X6.2: logic expressivity). The experiments vary the number of constraint clauses from 1 to 1500 and evaluate the model's search for global minima (X3.1.1.1: SAT solving) (Feng and Sathasivam 2024, see Abstract). However, the abstract does not establish support for either SAT or UNSAT decision or satisfiability witness generation and reports no assurance mechanism corresponding to the Y dimension.

The DHNN-2SAT-CSAFC framework embeds PL formulas written in CNF, with exactly two literals per clause, in a DHNN to solve 2SAT (X6.2: logic expressivity). The Crow Search Algorithm (CSA) guides fuzzy clustering in the context of global optimization. During retrieval, the method selects a subset of initial assignments with favorable cost functions, thereby narrowing the search space (X4.3: search-space pruning; X4.5: heuristic optimization for SAT). The DHNN retrieves global-minimum neuron states corresponding to consistent interpretations that satisfy every clause (X2.1: satisfiability witness generation; X3.1.1.1: SAT solving). The paper provides insufficient evidence to assign a Y-dimension assurance annotation. (Feng et al. 2024, pp. 9242–9244, Sections 3.1–3.2; p. 9244, Figure 5).

The study by Gao et al. (2024) extends the DHNN-GRAN3SAT framework introduced in Gao et al. (2022) by introducing BOSACO and BOSACO-t to optimize the retrieval phase (Gao et al. 2024, see Abstract). In (Gao et al. 2024, Sections 2–3), the G-Type Random 3 Satisfiability (GRAN3SAT) logic structure is embedded in a DHNN. GRAN3SAT represents a restricted class of PL formulas in CNF in which each clause contains one, two, or three literals (X6.2: logic expressivity) (Gao et al. 2024, Section 2.1). During the DHNN learning phase, the Binary Optimal Solution Ant Colony Optimization (BOSACO) optimizes the performance of the GRAN3SAT structure embedded in the DHNN. The DHNN then uses energy minimization to retrieve candidate neuron-state vectors (X4.5: heuristic optimization for SAT) (Gao et al. 2024, Sections 2–3). A zero value of the GRAN3SAT cost function indicates that the corresponding interpretation satisfies PL formula (X2.1: satisfiability witness generation; X3.1.1.1: SAT solving) (Gao et al. 2024, Section 2.1).

MTS-PRO2SAT embeds the Probabilistic 2 Satisfiability (PRO2SAT) logical structure into a DHNN. PRO2SAT is expressed in CNF, with clauses containing positive or negative literals (X6.2: logic expressivity). During the learning phase, the Hybrid Mutation Tabu Search (MTS) algorithm searches for a neuron-state assignment that minimizes the PRO2SAT cost function (X4.5: heuristic optimization for SAT). The tabu table records previously visited neuron-state assignments and helps prevent the MTS algorithm from revisiting them, thereby reducing redundant searches (X4.3: search-space pruning). The paper does not report an assurance mechanism corresponding to the Y dimension. Furthermore, no Y concept is assigned to the MTS-PRO2SAT framework (Chen et al. 2024, p. 3; pp.5–11, Sections3–6.1; p.17, Figure3).

TG-SAT encodes SAT instances represented in CNF as undirected graphs (Chang et al. 2024, p. 6, Figure 1). That graph is integrated into GRU-NN (X6.2: logic expressivity). The TG-SAT model is trained as a classifier to predict the SAT of a PL formula written in CNF (X1.1: satisfiability decision; X3.1.1.1: SAT solving). It is tested with instances having 150–300 variables (X5.1: logical generalization) (Chang et al. 2024, Table 5). The binary classification output is a satisfiability decision rather than constructive evidence. TG-SAT returns neither a satisfying assignment for a SAT prediction nor an UNSAT refutation for an UNSAT prediction (Chang et al. 2024, pp. 18–20). Accordingly, no Y -dimension annotation is assigned.

NeuroDual integrates a GAT NN into MiniSAT’s CDCL process to determine the satisfiability of PL formulas written in CNF (X6.2: logic expressivity). The GAT computes an assignment score for each variable, thereby providing a decision heuristically (X4.4: branching heuristic; X4.5: heuristic optimization for SAT). MiniSAT then performs the satisfiability search (X1.1: satisfiability

decision; X3.1.1.1: SAT solving). The accessible abstract reports neither satisfying assignment generation nor UNSAT refutation and describes no assurance mechanism corresponding to the Y dimension. Therefore, no Y concept is assigned to the NeuroDual framework (Farooque et al. 2024, see Abstract).

The paper Hůla et al. (2024) explicitly applies GNN to enhance NeuroSAT architecture to solve SAT in PL. NeuroSAT encodes a PL formula as graphs (X6.2: logic expressivity). NeuroSAT predicts whether the PL formula is SAT or UNSAT (X1.1: satisfiability decision; X3.1.1.1: SAT solving; X2.1: satisfiability witness generation). Applying decimation increases the percentage of SAT instances for which NeuroSAT finds a satisfying assignment (X4.3: search-space pruning; X4.5: heuristic optimization for SAT). Experiments using MaxSAT training support X3.2: MaxSAT optimization (Hůla et al. 2024, Introduction). Tests on larger Latin squares and Sudoku instances support X5.1: logical generalization (Hůla et al. 2024, Table 1; Section 5.3). None of the reported mechanisms corresponds to a concept in the Y dimension.

Graph Neural Network SAT Solver Selector (GraSS) combines a GNN with expert knowledge for per-instance SAT solver selection (Zhang et al. 2024b, p. 6301). It directly guides SAT solving (X3.1.1.1: SAT solving) and adapts solver choice per instance (X3.1.2.4: strategy adaptation), supporting decisions through the selected solver (X1.1: satisfiability decision). The GNN does not itself derive assignments or refutations. No Y concept is assigned because certificate generation, proof validation, and external verification are not described.

In Lymperopoulos and Liu (2024), the authors train a GNN on a graph representation of a CNF formula and use its output to prune clauses before an existing minimal unsatisfiable subsets (MUS) enumerator is applied. The learned component directly reduces the MUS search space (X4.3: search-space pruning). The symbolic enumerator performs satisfiability checks for SAT for CNF formulas (X1.1: satisfiability decision; X3.1.1.1: SAT solving). No proof-certificate mechanism is reported.

The framework presented in Guo et al. (2024) extends the YRAN2SAT-based work of Guo et al. (2022) by optimizing the learning and retrieval phases using the Hybrid Differential Evolution Algorithm (HDEA) and Swarm Mutation, respectively (Guo et al. 2024, see Abstract). YRAN2SAT defines a restricted class of PL formulas comprising conjunctions of randomly generated PL clauses (X6.2: logic expressivity) (Guo et al. 2024, pp. 14–15, Section 3). The framework embeds YRAN2SAT as a logical rule in a DHNN. During the DHNN learning phase, HDEA obtains neuron states that minimize the YRAN2SAT cost function (X4.5: heuristic optimization for SAT). The neuron

states minimizing this cost function represent truth interpretations that satisfy the SAT formula (X3.1.1.1: SAT solving) (Guo et al. 2024, p. 3, Section 1).

DeepSAT is a research framework that aims to apply RNN to automated theorem proving in PL (Amato et al. (2025)). A PL formula is initially converted into a binary syntax tree and then processed by a Tree-LSTM model. This Tree-LSTM model is trained to predict whether such a PL formula is a tautology (Amato et al. 2025, see Abstract). This trained model is integrated into a Monte Carlo Tree Search (MCTS) agent (Amato et al. 2025, p. 6, Section 4). The DeepSAT framework constructs a formal proof of validity (Y1.2: proof validation) (Amato et al. 2025, see Abstract).

The SOM-NN solver addresses the SAT problem in PL by utilizing a multi-step division method (X3.1.1.1: SAT solving) (Yun and Wang 2025, p. 6, Figure 3; p. 7, Figure 4). It includes preliminary clustering, SOM-NN clustering, and minimum cut division (Yun and Wang 2025, p. 7, Figure 4). The study does not document an assurance mechanism covered by the Y dimension Yun and Wang (2025).

LagONN uses phase-based oscillatory neurons and Lagrange oscillators to search for solutions to SAT for CNF formulas (X3.1.1.1: SAT solving) (Delacour et al. 2025, Sections 3.1–3.4). Successful runs find satisfying assignments for random SAT instances with up to 200 variables and 860 clauses (X2.1: satisfiability witness generation). Failure does not establish unsatisfiability.

The MILP-SAT-GNN model predicts the SAT of a PL formula expressed in CNF (3.1.1.1: SAT solving) Cardillo et al. (2025). Formulas expressed in CNF are mapped to MILP problems, which are then encoded in GNNs for training.

HyperSAT is a novel approach that employs a HyperGCN to solve the weighted MaxSAT problem (X3.2: MaxSAT optimization; see Table 32) (Chen et al. (2025)). HyperSAT uses a hypergraph to represent a weighted MaxSAT instance (Chen et al. 2025, p. 3, Figure 1). Rather than treating the MaxSAT problem as a formal reasoning task in PL, HyperSAT treats the prediction of solutions to the Weighted MaxSAT problem as an inference task over structured input (Chen et al. 2025, p. 2, the end of Section 1).

Liu et al. (2025b) define weighted random 3SAT as a CNF formula in which each clause contains three literals and encode this formula in a DHNN (X3.1.1.1: SAT solving; X4.5: heuristic optimization for SAT). The final neuron states encode truth assignments. Cost function was derived by minimizing the inconsistency of the logic (Liu et al. 2025b, see Abstract). Because the paper reports no assurance mechanism corresponding to the Y dimension, the framework receives an empty Y annotation set.

Table 32. X–Y annotations and ANN architectures of NeSy frameworks, 2025–2026.

ID	Reference	X and Y concept	ANN architecture
1	Amato et al. (2025)	Y1.2	Tree-LSTM NN
2	Mojzisek et al. (2025)	X1.1	GNN
3	Langedijk et al. (2025)	X2.1 X3.1.1.1 X5.1	LSTM, GCNN, Transformer
4	Chang and Liu (2025)	X1.1, X3.1.1.1 X5.1	GNN (SAT-GATv2)
5	Chen et al. (2025)	X3.2	HyperGCN
6	Chang et al. (2025)	X3.1.1.1	DHNN
7	Pan et al. (2025)	X1.1, X3.1.1.1 X4.3, X6.1, X6.2 X5.1, X7, Y3	Transformer
8	Liu et al. (2025b)	X3.1.1.1, X4.5	DHNN
9	Chen and Sathasivam (2025)	X3.1.1.1, X4.5	DHNN
10	Moriyama and Inoue (2025)	X3.2, X3.1.1.1	SGAT, GNN
11	Zhai and Ge (2025)	X1.1, X5.1 X3.1.1.1 X3.1.2.4 X4.4, X4.5	GNN
12	Liu et al. (2025a)	X3.2, X4.5	GNN
13	Delacour et al. (2025)	X3.1.1.1, X2.1	LagONN
14	Cardillo et al. (2025)	X3.1.1.1	GNN
15	Yun and Wang (2025)	X3.1.1.1	SOM-NN
16	Xie and Sathasivam (2026) Alway et al. (2026) Chen and Sathasivam (2026)	X2.1 X3.1.1.1 X4.5	DHNN
17	Tönshoff and Grohe (2026)	X1.1, X2.1 X3.1.1.1 X4.4, X4.5	GNN
18	Zhang et al. (2026)	X4.4, X6.1 X2.1, X3.1.1.1 X5.1	Perceiver AR, ImitSAT
19	Peltonen and Wattenhofer (2026)	X3.1.1.1	GNN
20	Sun et al. (2026)	X3.1.1.1, X2.1, X2.2, Y1.1.2	DeepSeek-V3
21	Hou (2026)	X3.1.1.1, X4.4	MPNN
22	Ahsan et al. (2026)	X3.2, X3.1.1.1	Autoencoder
23	Erata et al. (2026)	X3.1.1.1, X4.4	Qwen3-4B

A novel closest-assignment supervision method is introduced to improve the training process for solving SAT problem in PL [Mojzisek et al. \(2025\)](#). A GNN architecture is applied to predict satisfying assignments for Boolean SAT instances (X1.1: satisfiability decision); see Table 32) in PL [Mojzisek et al. \(2025\)](#). The terms inference and reasoning are both used throughout the paper. However, the authors do not adopt these notations in a formal logic sense (see Sections 2.2 and Sections 2.3). Instead, inference is used in the machine learning sense, denoting the number of message passing iterations

beyond the training distribution. The authors use the term reasoning more descriptively to characterize the ANN's behavior as analogous to the continuous relaxation of MaxSAT (Mojzisek et al. 2025, see Abstract).

SAT-GATv2 is a GNN-based model that is designed to predict whether a PL formula is satisfiable Chang and Liu (2025). The solution is based on transforming a PL formula written in CNF into a literal-clause graph (X1.1: satisfiability decision; X3.1.1.1: SAT solving) Chang and Liu (2025). The paper provides limited empirical evidence of logical generalization (X5.1). The paper does not generate an UNSAT refutation. Another decoder-only Transformer has been implemented to solve the 3SAT problem in PL (X1.1: satisfiability decision; X3.1.1.1: SAT solving) Pan et al. (2025). Unit propagation may reduce some clauses to a single literal (X4.3: search-space pruning), while implementing DPLL within the Transformer model supports X6.1: algorithmic alignment and X6.2: logic expressivity. Logical generalization (X5.1) is limited (Pan et al. 2025, Section 6). The formal correctness proof for the constructed model supports soundness guarantee (X7) and soundness assessment (Y3) (Pan et al. 2025, Appendix C.8).

Discrete HNNs are used to implement random 2SAT rules in PL (Chang et al. 2025, Sections 2–3; Section 4). Authors use a binary artificial bee colony algorithm to optimize logic generation (Chang et al. 2025, see Abstract). In addition, the work in Chen and Sathasivam (2025) uses discrete HNNs to solve the random 3SAT problem (X3.1.1.1: SAT solving; X4.5: heuristic optimization for SAT). The paper does not provide a formal UNSAT refutation. However, no concepts from the Y dimension are assigned to this paper.

An advanced, neural-based reasoning approach was used to develop a SGAT as a novel GNN for solving the MaxSAT problem in PL (X3.2: MaxSAT optimization; X3.1.1.1: SAT solving) (Moriyama and Inoue 2025, Section 2, MaxSAT formulation; Section 3, graph-based attention; Figures 1–2). The local search solver uses the SGAT to continuously solve the given MaxSAT problem.

Zhai and Ge (Zhai and Ge 2025, pp. 4–6, Sections 3.1–3.3, Sections 4.1–4.3; p. 16, Appendix B.2, Algorithm 1) use reinforcement learning to optimize splitting heuristics (X3.1.2.4: strategy adaptation; X4.4: branching heuristic; X4.5: heuristic optimization for SAT). Symbolic solvers, not the policy, decide satisfiability (X1.1: satisfiability decision; X3.1.1.1: SAT solving). The transfer from randomly generated 3SAT training instances to SAT Competition 2023 instances supports X5.1: logical generalization (Zhai and Ge 2025, p. 6, Section 4.4; pp. 7–9, Sections 5.1–5.2).

In the work Langedijk et al. (2025), authors examine the generalization of three neural architectures: Transformer, GCNN, and LSTM in a PL reasoning task. The task requires models to generate satisfying assignments

for PL formulas, making it a structured and interpretable setting for studying compositionality (X2.1: satisfiability witness generation; X3.1.1.1: SAT solving) (Langedijk et al. 2025, Abstract; pp. 4–8, Sections 3–4). NeSy frameworks are tested on PL formulas using the Prop50 dataset (X5.1: logical generalization) (Langedijk et al. 2025, pp. 8–14, Sections 4.5–6). No Y-dimension assurance mechanism was identified, leaving the Y annotation set empty.

PMS is a generalization of Boolean SAT and MaxSAT (Liu et al. 2025a, see Abstract). An InitPMS is trained through GNN to generate an initial assignment for MaxSAT instances while distinguishing hard and soft clauses. InitPMS then supplies the assignment to a local search solver, which uses it as the starting point for optimization (X3.2: MaxSAT optimization; X4.5: heuristic optimization for SAT) (Liu et al. 2025a, see Section 2.1). The framework does not report SAT decision, UNSAT refutation, or proof validation. No assurance mechanism represented in the Y dimension is reported.

As discussed in Peltonen and Wattenhofer (2026), GNNs are used as the main architecture for solving SAT problems. The authors investigate the expressive power of GNNs for solving the SAT problem in PL using the Weisfeiler–Lehman (WL) test (X3.1.1.1: SAT solving). They study the expressivity of GNNs on regular, random, and planar SAT instances. They report that random instances can be distinguished, while industrial instances require greater expressivity to predict a satisfying assignment (Peltonen and Wattenhofer 2026, see Abstract).

Three frameworks Xie and Sathasivam (2026); Alway et al. (2026); Chen and Sathasivam (2026) use discrete HNN to search for network states representing truth assignments that satisfy every clause of a PL formula in CNF (X2.1: satisfiability witness generation; X3.1.1.1: SAT solving). The frameworks employ distinct search procedures. Hybrid exhaustive search applies intelligent mutation (Alway et al. 2026, Sections 2–3), (K)-medoids clustering organizes candidate states before applying evolutionary learning (Chen and Sathasivam 2026, Sections 2–3), and clause-derived energy functions optimize network weights (Xie and Sathasivam 2026, Sections 2–3). These modifications improve the search for network states encoding truth assignments that satisfy every clause of the CNF formula (X4.5: heuristic optimization for SAT). However, failure to find such a network state neither establishes that the formula is UNSAT (X1.1) nor produces an UNSAT refutation (X2.2). The frameworks optimize search objectives, energy, or weights, but none seeks a truth assignment that maximizes the number or total weight of satisfied clauses for UNSAT formulas. This excludes them from MaxSAT optimization (X3.2).

In the paper (Tönshoff and Grohe 2026, Sections 3–4), the authors use a single GNN evaluation to calculate initial scores for the branching heuristic of a symbolic SAT solver. The symbolic solver returns either SAT or UNSAT (X1.1: satisfiability decision; X3.1.1.1: SAT solving). For a SAT result, the method finds a satisfying assignment for a PL formula written in CNF (X2.1: satisfiability witness generation). Neural initialization of variable scores supports the branching heuristic (X4.4) and heuristic optimization for SAT (X4.5), but the neural component does not independently determine SAT or construct an UNSAT refutation.

ImitSAT uses CDCL and KeyTrace to predict the next signed variable as the branch decision (Zhang et al. 2026, Introduction, Figure 1) (X4.4: branching heuristic; X6.1: algorithmic alignment). The CDCL solver decides SAT for CNF formulas (Zhang et al. 2026, Section 3, pp. 3–4) (X3.1.1.1: SAT solving), obtaining a satisfying assignment when the formula is satisfiable (X2.1: satisfiability witness generation) (Zhang et al. 2026, Section 3, pp. 3–4) or establishing UNSAT otherwise (Zhang et al. 2026, Section 3, p. 4; Algorithm 1, p. 14). Tests support generalization (X5.1: logical generalization) (Zhang et al. 2026, Section 5.3 and Table 2, p. 9) (Zhang et al. 2026, Section 5.5, p. 10; Appendix F, p. 17; Appendix Q, p. 22).

The AutoModSAT model uses the DeepSeek-V3 heuristics to optimize a CDCL solver for determining the satisfiability of a PL formula expressed in CNF (X3.1.1.1: SAT solving) (Sun et al. 2026, Methods). In the SAT case, the solver returns a truth assignment that satisfies the PL formula (X2.1: satisfiability witness generation). In the UNSAT case, the solver outputs a DRAT proof of unsatisfiability (X2.2: UNSAT refutation). Assignments are checked independently, and drat-trim verifies proofs (Y1.1.2: external verification) (Sun et al. 2026, Heuristics discovery section, p. 9).

The GMTSAT model guides Kissat, a CDCL solver, in determining the satisfiability of a PL formula expressed in CNF (X3.1.1.1: SAT solving). Its guidance strategy uses high-confidence GMTSAT predictions while maintaining the logical completeness of SAT solving (Hou 2026, p.85, Section4.5). In addition, the conservative guidance strategy injects neural predictions as a branching heuristic and achieves a zero error rate (X4.4: branching heuristic) (Hou 2026, p.90, Section5).

In Ahsan et al. (2026), the authors use an autoencoder-based neuromorphic architecture with Fowler–Nordheim annealing to maximize the number of satisfied clauses (X3.2: MaxSAT optimization; X3.1.1.1: SAT solving) Ahsan et al. (2026). No reported assurance mechanism satisfies the criteria for annotation under *Y*.

The Qwen3-4B-SFT-DPO, a neural cubing heuristic, guides Cube-and-Conquer (C&C) in addressing the SAT problem in PL (X3.1.1.1: SAT solving;

X4.4: branching heuristic) Erata et al. (2026). During cubing phase, the model selects splitting variables to partition the formula into subproblems. During conquering phase, a CDCL solver solves these subproblems (Erata et al. 2026, p. 2, Figure 1). The paper’s description does not support assigning an assurance concept from the Y dimension. The authors collected 94 SAT and 314 UNSAT formulas with at most 600 variables and 1000 clauses, selecting subsets for training. Evaluation used a separate test set of 50 SAT and 50 UNSAT formulas excluded from training (Erata et al. 2026, p. 4, Section 4.1).

6 Evaluation of Existing Work

This section evaluates the reported capability and assurance profiles of NeSy frameworks in 181 papers surveyed in Section 5, excluding McCulloch and Pitts (1943), which is retained for historical context. The X and Y dimensions of the X-Y-Z taxonomy guide paper annotation, while the Z dimension supports quantitative comparison. Concept codes are taxonomic notations, not numerical scores.

Subsection 6.1 defines $F(p) = (Z_1(p), Z_2(p), Z_3(p))$ and outlines the automated evaluation pipeline. Capability (Z_1) and assurance (Z_2) scores combine normalized frequency-weighted concept sums with normalized coverage. Joint coverage (Z_3) is the scaled harmonic mean of normalized X and Y coverage. All three scores lie in $[0, 6]$. Frequency counts papers annotated with a concept, whereas coverage counts distinct concepts assigned to a paper within each dimension. Subsection 6.2 describes data cleaning and construction of the canonical publication and concept-assignment tables.

Subsection 6.3 reports concept frequencies and weights, distinguishing annotation prevalence from scientific importance and highlighting unsigned concepts. Subsection 6.4 summarizes ANN architecture frequencies across evaluated publications. Finally, Subsection 6.5 examines yearly means, medians, and binned Shannon entropy of the three scores alongside publication counts, characterizing temporal variation in score levels and diversity across score intervals. These statistics describe annotation-based profiles rather than directly measuring framework quality or establishing soundness.

6.1 Formal Definition of the Evaluation Function

Let P be the set of publications included in the quantitative evaluation, excluding McCulloch and Pitts (1943), and let $N = |P|$ be the total number of surveyed papers. For a paper $p \in P$, let X_p be the set of X concepts assigned to paper p , and let Y_p be the set of Y concepts assigned to paper p . Thus, $|X_p|$ is the number of X concepts assigned to p , and $|Y_p|$ is the

number of Y concepts assigned to p . The evaluation pipeline computes, for each publication p , the following vector-valued function:

$$F(p) = (Z_1(p), Z_2(p), Z_3(p)). \quad (10)$$

The components are calculated using the definitions below. For a concept x from the X taxonomy shown in Figure 3, let $XP(x)$ denote the frequency of x , i.e., the number of surveyed papers annotated with x :

$$XP(x) = |\{p \in P \mid x \in X_p\}|. \quad (11)$$

Let $XW(x)$ denote the weight of concept x , computed as the proportion of surveyed papers annotated with x :

$$XW(x) = \frac{XP(x)}{N}. \quad (12)$$

Let $XR(p)$ denote the raw capability score of paper p , computed by summing the weights of all x concepts assigned to p :

$$XR(p) = \sum_{x \in X_p} XW(x). \quad (13)$$

Let $XC_{\max}$ denote the maximum number of different x concepts assigned to each paper in P :

$$XC_{\max} = \max_{p \in P} |X_p|. \quad (14)$$

Let $CX(p)$ denote the normalized X -coverage of paper p . This value ranges from 0 to 1 and expresses the relative breadth of the capability profile of p :

$$CX(p) = \begin{cases} \frac{|X_p|}{XC_{\max}}, & XC_{\max} > 0, \\ 0, & XC_{\max} = 0, \end{cases} \quad (15)$$

Let $XR_{\max}$ denote the maximum raw x score among all surveyed papers:

$$XR_{\max} = \max_{p \in P} XR(p). \quad (16)$$

Let $S = 6$ be the scaling constant that maps the final value to the range from 0 to 6. The final capability score of paper p is then:

$$Z_1(p) = \begin{cases} S \cdot \frac{XR(p)}{XR_{\max}} CX(p), & XR_{\max} > 0, \\ 0, & XR_{\max} = 0. \end{cases} \quad (17)$$

The assurance score is computed analogously. For a concept y from the Y taxonomy shown in Figure 4, let $YP(y)$ denote the frequency of y , i.e., the number of surveyed papers annotated with y :

$$YP(y) = |\{p \in P \mid y \in Y_p\}|. \quad (18)$$

Let $YW(y)$ denote the weight of concept y , computed as the proportion of surveyed papers annotated with y :

$$YW(y) = \frac{YP(y)}{N}. \quad (19)$$

Let $YR(p)$ denote the raw assurance score of paper p , computed by summing the weights of all y concepts assigned to p :

$$YR(p) = \sum_{y \in Y_p} YW(y). \quad (20)$$

Let $YC_{\max}$ denote the maximum number of y concepts assigned to each paper in P :

$$YC_{\max} = \max_{p \in P} |Y_p|. \quad (21)$$

Let $CY(p)$ denote the normalized y -coverage of paper p :

$$CY(p) = \begin{cases} \frac{|Y_p|}{YC_{\max}}, & YC_{\max} > 0, \\ 0, & YC_{\max} = 0, \end{cases} \quad (22)$$

Let $YR_{\max}$ denote the maximum raw y score among all surveyed papers:

$$YR_{\max} = \max_{p \in P} YR(p). \quad (23)$$

The final assurance score of paper p is then:

$$Z_2(p) = \begin{cases} S \cdot \frac{YR(p)}{YR_{\max}} CY(p), & YR_{\max} > 0, \\ 0, & YR_{\max} = 0. \end{cases} \quad (24)$$

Thus, Z_1 and Z_2 are not computed from the syntax of the concept identifier. Instead, they are computed from the concepts assigned to a paper, how common these concepts are in the surveyed corpus, and how many distinct concepts are used to annotate the paper.

The joint coverage score $Z_3(p)$ is the harmonic mean of the normalized coverage values $CX(p)$ and $CY(p)$, scaled by $S = 6$:

$$Z_3(p) = \begin{cases} S \cdot \frac{2 \text{CX}(p) \text{CY}(p)}{\text{CX}(p) + \text{CY}(p)}, & \text{if } \text{CX}(p) + \text{CY}(p) > 0, \\ 0, & \text{otherwise.} \end{cases} \quad (25)$$

It is not the harmonic mean of $Z_1(p)$ and $Z_2(p)$. The harmonic mean gives greater influence to the smaller coverage value, so a high score requires substantial coverage in both dimensions. The score ranges from 0 to 6: it is 0 when either coverage value is 0 and reaches 6 only when both coverage values equal 1. If no surveyed paper has concepts assigned in a dimension, its normalized coverage is defined as 0 for all papers to avoid division by zero.

The evaluation pipeline is fully automated and implemented as a Python project, which is hosted in a GitLab repository. The repository URL is provided in Listing 1 of Appendix A.1. Figure 8 illustrates the overall evaluation pipeline, which consists of five steps.

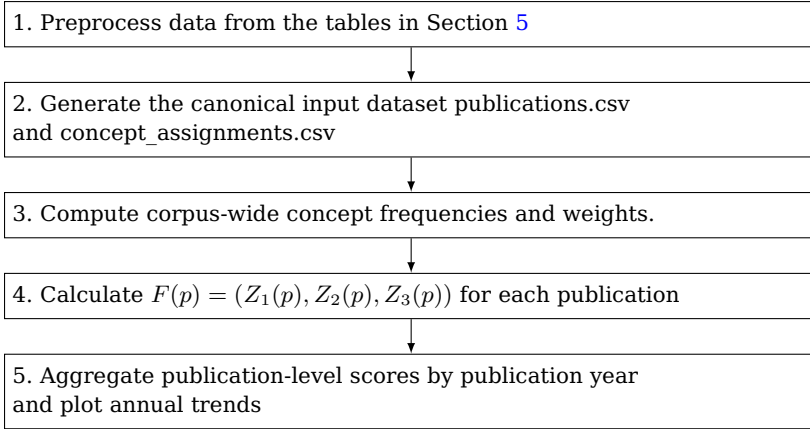


Figure 8. Evaluation pipeline implemented in the nesy_evaluation Python project (see Listing 1 in Appendix A.1).

6.2 Data Preprocessing

In the pre-processing phase, the evaluation pipeline takes as input the data presented in Tables 23–32 in Section 5. Figure 8 illustrates preprocessing and canonical dataset generation as the two initial steps preparing data for computing $F(p) = (Z_1(p), Z_2(p), Z_3(p))$. OpenAI’s GPT-6-Astra is used as a tool to extract and organise the data. Continuation rows are combined, shared entries are expanded into separate publication records, identified formatting errors are corrected, and duplicate concept assignments are removed. No unstated parent concepts are added. Publication years are extracted from

citation keys and require bibliographic verification before temporal analysis. The paper [McCulloch and Pitts \(1943\)](#) is retained for historical context but excluded from the canonical dataset and quantitative evaluation. In the second step, the extracted data are organized into the canonical input dataset. The dataset comprises two tables:

The `publications.csv` table contains four fields: `paper_id`, `citation_key`, `publication_year`, and `ANN_architectures`. Each citation key receives one unique paper ID and occupies one row, with multiple ANN architecture names separated by semicolons within the same cell. The `concept_assignments.csv` table contains three fields: `paper_id`, `dimension`, and `concept_id`. Each distinct paper–concept pair occupies a separate row, linked to an existing publication through its paper ID, establishing a one-to-many relationship between publications and assignments. The table contains 525 concept assignments: 512 for X concepts and 13 for Y concepts. There are 10 publications with multiple ANN architectures.

Both files are stored in the `data/raw` folder of the `nesy_evaluation` Python project, whose URL is provided in Listing 1 of Appendix A.1. They provide the input data for steps 3–5 in Figure 8.

6.3 Concept Frequencies and Weights

Using the dataset generated in step 2, step 3 computes the frequencies and weights of concepts from the X and Y dimensions across the 180 evaluated publications. The results are stored in the `data/processed` folder of the `nesy_evaluation` Python project and are reported in Table 33.

Table 33 shows that SAT solving (X3.1.1.1) and logic expressivity (X6.2) are the most frequently assigned concepts, appearing in 117 and 112 publications, respectively. Their weights, approximately 0,650 and 0,622, represent their proportions in the evaluated corpus. Satisfiability witness generation (X2.1) occurs in 69 publications, whereas UNSAT refutation (X2.2) occurs in only 2 [Ghanem \(2024\)](#). In the assurance dimension, soundness assessment (Y3) occurs in 7 publications and proof validation (Y1.2) in 2. These frequencies describe annotation prevalence rather than measured effectiveness or the strength of a formal guarantee. Under the defined formulas, more frequent concepts contribute greater weights to the raw scores (13) and (20).

Including the root and intermediate nodes, the supplied taxonomies contain 36 X concepts and 11 Y concepts. Of these, 23 X concepts and 6 Y concepts appear in the annotations, leaving 13 and 5 concepts, respectively, without explicit assignments. However, these totals include unassigned organizational nodes because parent concepts are not automatically added when their descendants are assigned. Considering leaf concepts alone, 3

Table 33. Corpus-wide frequencies and weights of *X* and *Y* concepts across 180 evaluated publications. Frequency denotes the number of publications assigned a concept. Weight is its frequency divided by 180, rounded to three decimal places.

ID	Concept ID	Paper frequency	Concept weight
1	X1.1: Satisfiability decision	20	0.111
2	X1.2: Logical consequence	17	0.094
3	X2.1: Satisfiability witness generation	69	0.383
4	X2.2: UNSAT refutation	2	0.011
5	X3.1.1.1: SAT solving	117	0.650
6	X3.1.1.2: Resolution	4	0.022
7	X3.1.2: Heuristic adaptation	1	0.006
8	X3.1.2.2: RL-guided clause selection	1	0.006
9	X3.1.2.3: Query-aware formula generation	2	0.011
10	X3.1.2.4: Strategy adaptation	3	0.017
11	X3.2: MaxSAT optimization	21	0.117
12	X4.2: Clause selection quality	1	0.006
13	X4.3: Search-space pruning	8	0.044
14	X4.4: Branching heuristic	11	0.061
15	X4.5: Heuristic optimization for SAT	64	0.356
16	X5.1: Logical generalization	12	0.067
17	X5.2: Paradox-resistant training	1	0.006
18	X5.3: UNSAT-core prediction	2	0.011
19	X6.1: Algorithmic alignment	28	0.156
20	X6.2: Logic expressivity	112	0.622
21	X7: Soundness guarantee	10	0.056
22	X8.1: Robustness	4	0.022
23	X8.2: Interpretability	2	0.011
24	Y1.1: Certificate support	1	0.006
25	Y1.1.1: Certificate format	1	0.006
26	Y1.1.2: External verification	1	0.006
27	Y1.2: Proof validation	2	0.011
28	Y1.4.1: UNSAT certification	1	0.006
29	Y3: Soundness assessment	7	0.039

X concepts and 2 *Y* concepts remain unassigned: strategy monitoring (X3.1.2.1), lemma discovery (X4.1), lemma reuse (X4.6), proof compression (Y1.3), and interactive explanation (Y2). These concepts have zero frequency and weight and are omitted from the table of observed concepts.

For readers, this distinction helps separate taxonomic organization from potentially under-explored research topics. Unassigned leaf concepts identify properties for which the evaluated corpus contains no explicit annotations, suggesting priorities for closer literature review and framework analysis. They do not establish that these properties are absent from all NeSy frameworks. However, missing assignments may also reflect limited

reporting, annotation choices, or survey coverage. Similarly, a low weight does not imply low scientific importance.

6.4 Distribution of ANN Architectures Across Surveyed Publications

Table 34 presents ANN architecture frequencies, where frequency denotes the number of publications assigned an ANN architecture label. These data were generated using the nesy_evaluation Python project (see Listing 1 in Appendix A.1).

Table 34. Corpus-wide frequencies of ANN architecture labels across 180 evaluated publications. Labels sharing the same frequency are grouped in one row; the frequency applies to each label individually. Publications with multiple labels contribute to each corresponding count.

ID	ANN architectures	Paper frequency
1	ANN as directed graph, ANN for end-to-end diff. proving, Analog ANN, Autoencoder, BM, BPNN, Cont. time ANN, DeepSeek-V3, Elliot HNN, GAT, GCNN, GGC-NN, GM5-E, GM5-N, GNN (SAT-GATv2), GRU-NN, GraSS, HT ANN, HyperGCN, INN, ImitSAT, LBM, LINN, LNN, LSTM, LSTM RNN, LagONN, MHNN, NeuroSAT, Perceiver AR, PossibleWorldNet, Qwen3-4B, RBM, SAANN, SATNet, SGAT, SOM-NN, TenSAT, Tree-LSTM NN, Tree-NN, motivated by HNN, multi-layer	1
2	BERT, ConvNN, DNN, KBANN, Little HNN, RBFNN, RNN, SNN, Transformer, UANN	2
3	MPNN, higher order HNN	3
4	LPPH	4
5	FNN	8
6	ConnectNN	9
7	GNN	27
8	DHNN	31
9	HNN	42

ANN usage across the surveyed publications is concentrated in a few architecture categories. HNN is the most common label, appearing in 42 publications, followed by DHNN in 31 and GNN in 27. ConnectNN and FNN appear in 9 and 8 publications, respectively, while LPPH appears in 4. MPNN and higher order HNN each appear in 3 publications. Ten other categories each appear in two publications, while the remaining 42 each appear in one. These counts highlight the prominence of HNN, DHNN, and GNN alongside substantial diversity among less common architectures.

6.5 Mean, Median, and Shannon Entropy of Z_1 , Z_2 , and Z_3

In Steps 4 and 5 of the evaluation pipeline (see Figure 8), the nesy_evaluation Python project (see Listing 1 in Appendix A.1) first computes $F(p) = (Z_1(p), Z_2(p), Z_3(p))$ for each evaluated publication p . It then groups these publication-level scores by publication year and computes the following summary statistics separately for Z_1 , Z_2 , and Z_3 : the mean (Montgomery and Runger 2014, p. 190, Section 6.1), median (Montgomery and Runger 2014, p. 223, Example 7.1; p. 200, Example 6.5), and binned Shannon entropy (Shannon 1948, p. 394).

The arithmetic mean describes the average score among publications in a given year, whereas the median identifies the middle value of the ordered scores, or the arithmetic mean of the two middle values when the number of scores is even, and is less sensitive to extreme values. Shannon entropy complements the mean and median by summarising the distribution of publication-level scores within each year. For its calculation, scores are grouped into the fixed intervals $[0, 1)$, $[1, 2)$, $[2, 3)$, $[3, 4)$, $[4, 5)$, and $[5, 6]$, and the proportion of publications in each interval defines its probability. This calculation is performed separately for Z_1 , Z_2 , and Z_3 . These score intervals are used in the calculation but are not displayed in Figure 11, which shows the resulting yearly entropy values in bits. Entropy reflects both the number of occupied score intervals and how evenly publications are distributed among them; higher entropy does not imply higher capability or assurance. Figures 9, 10, and 11 present these statistics against actual publication years, allowing temporal variation to be examined independently of the boundaries chosen for chronological periods. The periods G5–G9 are retained only as visual annotations, and yearly publication counts provide context for interpreting statistics based on small samples. Years without evaluated publications have no summary statistics and appear as gaps in the statistical plots. All yearly results are stored in `z_statistics_by_year.csv` in the `data/processed/` folder of the `nesy_evaluation` project (see Listing 1 in Appendix A.1).

Figure 9 shows that the yearly mean of Z_1 fluctuates rather than increasing consistently over time. Its highest value is 3.872 in 2007, although this year contains only one evaluated publication. Other peaks occur in 2000 and 2005, both with a mean of 2.567, and in 2013, with a mean of 2.312. Within G9, the yearly mean increases from 0.798 in 2020 to 1.486 in 2021, decreases to 1.197 in 2023, and reaches 1.897 in 2024 before falling to 0.767 in 2025 and rising to 1.119 in 2026. These fluctuations illustrate the additional detail revealed by yearly aggregation that would be obscured by a single mean for each chronological period.

The yearly means of Z_2 and Z_3 are zero in most years with evaluated publications, with nonzero values occurring in selected years. Both reach their highest yearly means in 1999: 1.037 for Z_2 and 1.143 for Z_3 , based on three publications. Nonzero values occur in 1996, 2014, 2018, 2021, 2024, 2025, and 2026. For example, the respective means in 2018 are 0.667 and 0.645, whereas in 2024 they are 0.111 and 0.133. These values describe the scores obtained under the evaluation scheme and should not be interpreted as direct measurements of the overall quality of the publications.

The bottom panel shows that the number of evaluated publications varies substantially between years and is generally larger in the most recent period.

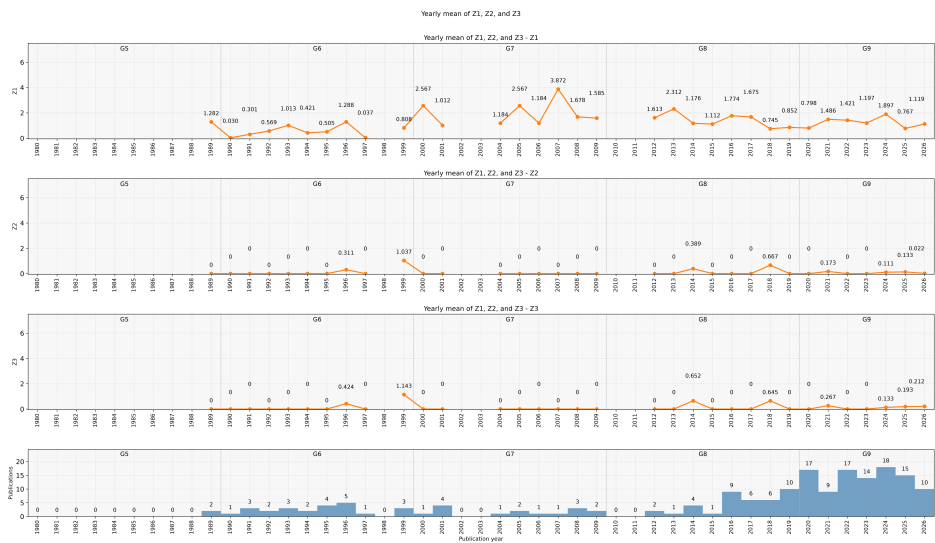


Figure 9. Yearly arithmetic means of Z_1 , Z_2 , and Z_3 across the evaluated publications. The bottom panel shows the number of evaluated publications per year. Chronological periods G5–G9 are annotated for reference.

The largest yearly count is 18 in 2024, while 2020 and 2022 each contain 17 publications. Consequently, the yearly means must be interpreted alongside their publication counts: a peak based on one publication, such as the Z_1 mean in 2007, provides less evidence of a broader pattern than a mean based on a larger set of publications. Overall, the figure does not show a sustained increase in all three evaluation metrics despite the larger number of evaluated publications in recent years.

Figure 10 shows that the yearly median of Z_1 fluctuates without a consistent upward trend. Its highest value is 3.872 in 2007, when only one publication was evaluated and the median therefore equals the mean. Other peaks occur in 2000 and 2005, both with a median of 2.567, and in 2013, with a median of 2.312. Within G9, the median increases from 0.697 in 2020 to 2.184 in 2021, decreases to 0.907 in 2023, and rises to 1.669 in 2024. It then falls to 0.489 in 2025 before increasing to 1.165 in 2026. These changes show that the central value of the Z_1 scores varies substantially between individual publication years.

The yearly medians of Z_2 and Z_3 are zero in every year with evaluated publications except 1999, when they reach 1.556 and 1.714, respectively, based on three publications. This pattern differs from the yearly means in Figure 9, which are positive in 1996, 2014, 2018, 2021, 2024, 2025, and 2026.

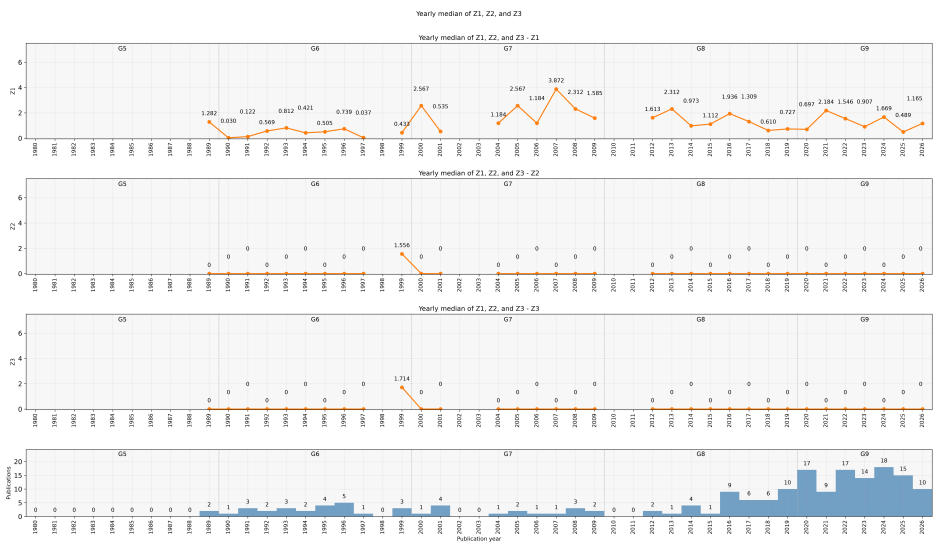


Figure 10. Yearly medians of Z_1 , Z_2 , and Z_3 across the evaluated publications. The bottom panel shows the number of evaluated publications per year. Chronological periods G5–G9 are annotated for reference.

Positive means alongside zero medians indicate that nonzero scores occur in a minority of the evaluated publications in those years. For example, the means of Z_2 and Z_3 in 2024 are 0.111 and 0.133, respectively, while both medians remain zero. Thus, the median complements the mean by showing that these positive scores do not characterise the central publication-level values.

The bottom panel provides the publication counts needed to interpret these medians. The largest count is 18 in 2024, followed by 17 publications in both 2020 and 2022, whereas several earlier years contain only one publication. Although the median is less sensitive to extreme scores than the mean, a median based on very few publications still provides limited evidence of a broader pattern. The larger publication counts in G9 coincide with fluctuating Z_1 medians and consistently zero Z_2 and Z_3 medians. These results describe the evaluated corpus under the adopted scoring scheme and do not establish a sustained improvement across all three metrics.

Figure 11 shows how publication-level scores are distributed across the fixed score intervals within each year. In plain terms, entropy indicates whether publications fall into one score interval or are distributed among several intervals. Zero entropy means that all scores belong to the same interval, although their exact values need not be identical or zero. The entropy of Z_1 varies considerably over time, reaching 1.371 bits in 1996 and 1.585 bits

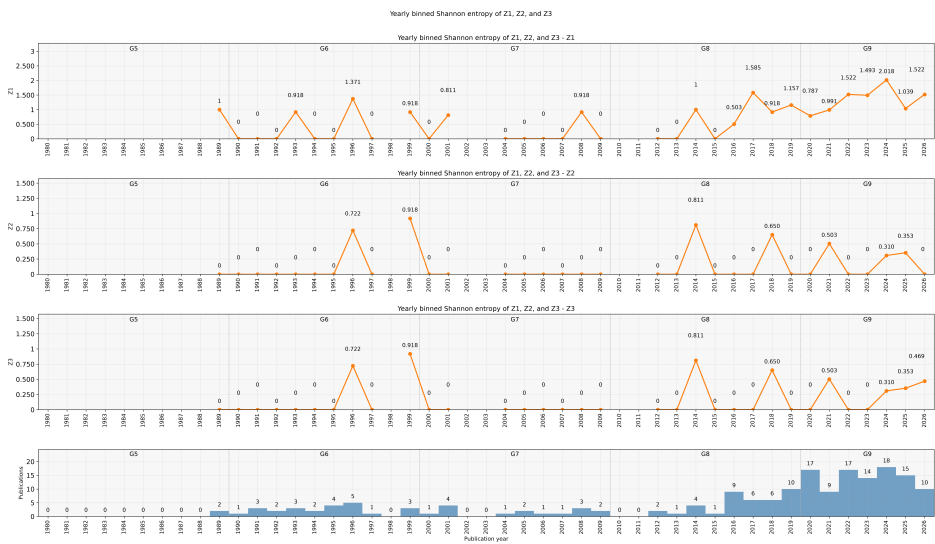


Figure 11. Yearly binned Shannon entropy of Z_1 , Z_2 , and Z_3 , measured in bits using fixed score intervals. The bottom panel shows the number of evaluated publications per year. Chronological periods G5–G9 are annotated for reference.

in 2017. Within G9, it increases from 0.787 bits in 2020 to its highest observed value of 2.018 bits in 2024, before decreasing to 1.039 bits in 2025 and rising to 1.522 bits in 2026. The peak in 2024 therefore indicates greater diversity across the score intervals, not the highest average capability score.

The entropy curves for Z_2 and Z_3 coincide except in 2026. Both reach their highest value of 0.918 bits in 1999, with further nonzero values of 0.722 bits in 1996, 0.811 bits in 2014, 0.650 bits in 2018, 0.503 bits in 2021, 0.310 bits in 2024, and 0.353 bits in 2025. In 2026, their entropies are 0 and 0.469 bits, respectively. These results indicate that the scores occupy multiple intervals only in selected years. Matching entropy values do not imply identical Z_2 and Z_3 scores, since different score distributions can produce the same entropy. The mean and median figures therefore remain necessary for interpreting the levels of these scores.

The publication counts in the bottom panel also matter when interpreting entropy. A year containing only one publication necessarily has zero empirical entropy, regardless of that publication’s score. For example, the highest yearly mean and median of Z_1 occur in 2007, but its entropy is zero because only one publication was evaluated. In comparison, the Z_1 entropy peak in 2024 is based on 18 publications. Differences in sample size and the selected score intervals should therefore be considered when comparing years.

Together, the three statistics distinguish average score, central score, and diversity across score intervals, without establishing that greater diversity represents better reasoning or stronger assurance.

Taken together, the statistics show that the surveyed frameworks exhibit capabilities represented by X concepts, while assurance represented by Y concepts is less consistently reflected in their annotations and scores. Positive yearly Z_1 means coexist with zero Z_2 medians in nearly every evaluated year, indicating that capability and assurance do not necessarily occur together. However, a zero assurance score should not be treated as proof that a framework is unsound; it indicates that the evaluation did not capture assurance through the assigned Y concepts. Students should therefore examine the underlying annotations and supporting evidence alongside the summary statistics. As a take away message: evidence of a NeSy framework's ability to perform reasoning or proving tasks does not, by itself, establish the correctness of its results. Readers should assess both the capabilities demonstrated and the assurance mechanisms reported, including the scope and limitations of any guarantees provided.

7 Conclusions

This survey examined PL inference, reasoning, and proving in NeSy frameworks through qualitative analysis and quantitative assessment. The main finding is an imbalance between the capabilities reported in the surveyed publications and the assurance mechanisms documented for their results. Across 181 publications discussed and 180 quantitatively evaluated, the literature provides substantial evidence of SAT-related capabilities, but much less consistent evidence of proof validation and other assurance mechanisms. Demonstrating that a framework performs a reasoning task therefore needs to be distinguished from establishing why its results should be trusted.

The comparison of 26 existing survey and review papers in Section 4 identifies a related gap in how the field is reviewed. Classifications based on architectures, knowledge representations, or learning methods do not consistently make explicit which reasoning and proving tasks in PL are supported and what evidence supports their correctness. The X-Y-Z taxonomy complements these perspectives by separating capability from assurance and making their reported coverage available for systematic comparison. Its hierarchy organizes concepts by specificity, not by increasing capability or assurance. The results in Section 6 make this imbalance concrete. Table 33 records SAT solving (X3.1.1.1) in 117 publications and satisfiability witness generation (X2.1) in 69, compared with proof validation (Y1.2) in 2 publications and soundness assessment (Y3) in 7. The yearly analysis

in Section 6.5 shows positive capability means alongside zero assurance medians in nearly every evaluated year. These findings support greater attention to proving and assurance in NeSy research. Explainability can help users understand results, but does not by itself establish logical validity or replace proof checking. The survey does not establish that an emphasis on explainability caused the observed imbalance.

For researchers, the practical implication is to assess each claim against the evidence appropriate to it. Publications should identify reasoning and proving tasks in PL, distinguish predictions from satisfying witnesses and proofs, and explain which component establishes correctness. When a neural component guides a symbolic solver, authors should state whether the guarantee comes from the solver, a proof checker, or another mechanism. Evaluation should report the assumptions and scope of guarantees alongside empirical performance, and make validation procedures and proof artifacts available where applicable. These distinctions also help students and reviewers examine capability and assurance claims consistently.

The quantitative results describe the curated corpus and its annotations. Concept frequency does not measure scientific importance, and the Z scores do not directly measure task performance or guarantee strength. A missing Y annotation or zero assurance score does not establish unsoundness. Reporting limitations, annotation choices, incomplete search coverage, and unequal yearly publication counts can affect the results. The findings therefore do not demonstrate a uniform decline across NeSy research. The Python pipeline makes calculations reproducible from the curated inputs, while publication screening and concept assignment still require scholarly judgment.

OpenAI’s GPT-6-Astra was prompted to conduct web searches for primary studies by publication period, using terminology variants, eligibility criteria, and deduplication rules (Appendix A.2). Surveys were retrieved separately through a documented DBLP SPARQL query. Each publication’s metadata and eligibility were manually checked. Bibliographic notes indicate cases where peer-review status could not be confirmed. Peer-review status was confirmed for all other publications. Searching public pages does not establish comprehensive database coverage. Manual database searches likewise cannot guarantee completeness because of coverage, indexing, and query limitations. Reference errors, omissions, and selection bias remain risks. Screening cannot establish completeness, and the revised prompt alone cannot ensure reproducibility.

Future work should develop and evaluate NeSy methods that produce independently checkable witnesses and proofs, and examine the cost of certification alongside accuracy, runtime, and scalability. The survey

methodology can be strengthened through independent annotation, assessment of annotator agreement, and sensitivity analysis of weighting and normalization choices. Updating the corpus and investigating unassigned concepts would help distinguish reporting gaps from underexplored capabilities. Extending the taxonomy to predicate, temporal, and other formal logics would require additional concepts and logic-specific assurance criteria. The released taxonomy, data, and implementation provide a basis for these extensions and for assessing future progress in both capability and assurance.

A Appendix

A.1 Supplementary Material

1. The `nesy_evaluation` Python project: https://gitlab.com/TIBHannover/human-decision/neuro-symbolic-reasoning-survey/-/tree/main/nesy_evaluation
2. SPARQL query: <https://gitlab.com/TIBHannover/human-decision/neuro-symbolic-reasoning-survey/-/tree/main/sparql>
3. Implemented X-Y-Z taxonomy as instance of SKOS ontology: <https://gitlab.com/TIBHannover/human-decision/neuro-symbolic-reasoning-survey/-/tree/main/taxonomy>

A.2 Prompt for the publication search

Transparency note. The following prompt was developed with assistance from OpenAI’s ChatGPT and used with OpenAI’s GPT-6-Astra to conduct web searches. Bibliographic information and eligibility were manually verified for each identified publication. Acceptance and peer-review status were not systematically verified for all papers accessed through arXiv or OpenReview. Manual assessment of scientific content should therefore be distinguished from verification of publication status.

Search objective. Conduct a systematic worldwide search for research papers, including peer-reviewed publications and publicly accessible preprints, describing neuro-symbolic (NeSy) frameworks that integrate neural and symbolic components to perform inference, reasoning, or proving specifically in propositional logic (PL).

Search independently within each of the following publication periods: 1940–1949, 1950–1959, 1960–1969, 1970–1979, 1980–1989, 1990–1999, 2000–2009, 2010–2019, and 2020–2026. Where a corresponding peer-reviewed publication is verified, use its publication year to assign the paper to a decade. Otherwise, use the year in which the preprint was first made publicly available and explicitly identify this basis for the assignment. Do

not require historical papers to use the modern terms “neuro-symbolic” or “NeSy.”

Mandatory inclusion criteria. Include a paper only when all the following criteria are satisfied:

- It is a full research paper available as a journal publication, a conference paper, or a publicly accessible preprint.
- It describes a framework or method containing both a neural component and an explicit symbolic component.
- Its symbolic representation substantively uses propositional logic, Boolean formulae, propositional calculus, CNF or DNF formulae, SAT, MaxSAT, propositional resolution, or another explicitly propositional inference or proof system.
- The interaction between its neural and symbolic components contributes directly to inference, reasoning, satisfiability solving, witness generation, refutation, proof search, or proving.
- Its bibliographic information can be verified from an authoritative venue, publisher, repository, or bibliographic record. Record acceptance or peer-review status separately where supporting evidence is available. If this status cannot be verified, state that explicitly rather than inferring it from the hosting platform.

A passing reference to propositional logic is insufficient. Propositional logic must be part of the framework’s representation, operation, inference procedure, reasoning mechanism, solver, evaluation task, or proof procedure.

Exclusion criteria. Exclude:

- Purely neural methods without a substantive symbolic component;
- Purely symbolic solvers or theorem provers without a neural component;
- Papers restricted to another logical formalism without a distinct and substantive PL component;
- Papers in which logic is used only as motivation, terminology, an output label, or a minor regularisation constraint;
- Surveys, tutorials, theses, posters, abstracts, presentations, non-research reports, and patents;
- Manuscripts for which the available text or bibliographic information is insufficient to assess eligibility; and
- Duplicate preprint, conference, journal, or technical-report versions of the same contribution.

Search coverage. Search publicly accessible records and pages associated with:

- publisher and conference websites;

- arXiv: <https://arxiv.org/> and other preprint repositories;
- OpenReview: <https://openreview.net/>;
- DBLP: <https://dblp.org/>;
- Semantic Scholar: <https://www.semanticscholar.org/>;
- Consensus: <https://consensus.app/>;
- OpenAlex: <https://openalex.org/>;
- Crossref: <https://www.crossref.org/>;
- Scopus: <https://www.scopus.com/> ;
- Web of Science: <https://www.webofscience.com/>;
- IEEE Xplore: <https://ieeexplore.ieee.org/>;
- ACM Digital Library: <https://dl.acm.org/>;
- Springer Nature Link: <https://link.springer.com/>;
- Springer Nature: <https://www.nature.com/>;
- university and institutional repositories;
- DOI records; and
- authors' institutional and publication pages.

Use combinations and spelling variants of terms such as “neuro-symbolic,” “neurosymbolic,” “neural-symbolic,” “connectionist-symbolic,” “neural logic,” “neural SAT,” “neural theorem proving,” and “differentiable logic,” together with PL-related terms such as “propositional logic,” “Boolean formula,” “CNF,” “SAT,” “UNSAT,” “MaxSAT,” “resolution,” “truth assignment,” “refutation,” and “propositional proof.”

Use arXiv, other preprint repositories, and OpenReview to discover candidates, access manuscripts, and identify corresponding publication versions. Include a preprint when it satisfies the substantive inclusion criteria, even if a peer-reviewed publication cannot be verified. Where a corresponding journal or conference publication is verified, cite that publication and record the manuscript version consulted. Otherwise, identify the work as a preprint or submission and state that acceptance or subsequent publication has not been verified. Availability on arXiv or OpenReview alone does not establish peer-review or acceptance status. Do not count a preprint and its published version as separate papers.

Comparison with the existing reference list. Compare every identified paper with the attached reference list or BibTeX file [ATTACHED REFERENCE FILE]. Detect papers already represented in the reference list by comparing normalised titles, DOI identifiers, arXiv identifiers, author lists, publication years, and preprint-publication relationships. Exclude all papers already represented in the attached reference list from the set of newly identified papers.

Required output. Provide:

- a table of newly identified eligible papers organised by decade;

- complete and verified bibliographic information for every new paper;
- a DOI or stable publication or repository URL;
- the publication or submission status of each paper, together with supporting evidence where available, or an explicit statement that acceptance or peer-review status has not been verified;
- evidence identifying its neural and symbolic components;
- evidence that propositional logic is used substantively;
- evidence that the framework performs inference, reasoning, or proving;
- a concise justification for inclusion or exclusion;
- a list of papers excluded because they are already present in the attached reference list; and
- the number of eligible new papers found in each decade. Report separately the number with verified peer-reviewed publication status and the number without such verification. Report zero when no eligible new paper is found; do not relax the criteria to populate a decade.

Create a downloadable BibTeX file named `new_nesy_pl_papers.bib`. This file must contain only newly identified papers satisfying all inclusion criteria. Do not include papers already represented in the attached reference list or alternative versions of those papers. Use only verified bibliographic fields and do not invent missing information, DOI identifiers, publication venues, or page numbers. For each paper accessed through arXiv, another preprint repository, or OpenReview, include a note field identifying the verified journal or conference publication, where available. Otherwise, identify the work as a preprint or submission and state that acceptance or subsequent publication has not been verified.

B Acknowledgements

This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation)–Projektnummer 543081196 (HUMAN-DECISION).

During the preparation of this manuscript, the authors used [GPT-6-Astra OpenAI](#) tool and [DeepL](#) to improve the readability of the text and correct grammatical errors. [GPT-6-Astra OpenAI](#) tool, including its web-search functionality, was also used to search for publications following the procedure described in Section 5, to represent the taxonomy as a SKOS ontology, and to generate an initial version of the Python code used in the evaluation workflow. The authors reviewed, adapted, and validated all AI-generated material and verified the bibliographic information and relevance of the publications identified through the search. The authors remain fully responsible for the scientific content, the selection and interpretation of the

literature, the accuracy of the ontology design and implementation, and the results reported in this paper.

The authors would like to thank reviewers for their valuable comments and suggestions.

References

- Abdechiri M and Meybodi MR (2012) A Hybrid Hopfield Network-Imperialist Competitive Algorithm for Solving the Satisfiability Problems. *International Journal of Computer and Electrical Engineering* 4(5): 726–732. DOI:10.7763/IJCEE.2012.V4.594. URL <https://doi.org/10.7763/IJCEE.2012.V4.594>.
- Abdeen S, Kasihmuddin MSM, Zamri NE, Manoharam G, Mansor MA and Alshehri N (2023) S-Type random k satisfiability logic in discrete Hopfield neural network using probability distribution: Performance optimization and analysis. *Mathematics* 11(4): 984. DOI:10.3390/math11040984. URL <https://doi.org/10.3390/math11040984>.
- Abubakar H (2022) An optimal representation of random maximum k -satisfiability on a Hopfield neural network for high order logic ($k \leq 3$). *Kuwait Journal of Science* 49(1). DOI:10.48129/kjs.11861. URL <https://doi.org/10.48129/kjs.11861>.
- Abubakar H and Danrimi ML (2021) Hopfield type of Artificial Neural Network via Election Algorithm as Heuristic Search method for Random Boolean kSatisfiability. *International Journal of Computing and Digital Systems* 10(1): 659–673. DOI:10.12785/ijcds/100163. URL <https://doi.org/10.12785/ijcds/100163>.
- Abubakar H, Masanawa AS, Yusuf S and Boaku GI (2021) Optimal representation to High Order Random Boolean kSatisfiability via Election Algorithm as Heuristic Search Approach in Hopeld Neural Networks. *Journal of the Nigerian Society of Physical Sciences* : 201–208 DOI:10.46481/jnsps.2021.217. URL <https://doi.org/10.46481/jnsps.2021.217>.
- Abubakar H, Masanawa SA and Yusuf S (2020a) Neuro-Symbolic Integration of Hopfield Neural Network for Optimal Maximum Random kSatisfiability (Maxrksat) Representation. *Journal of Reliability and Statistical Studies* 13(1): 199–220. DOI:10.13052/jrss1550-4646.13110. URL <https://doi.org/10.13052/jrss1550-4646.13110>.
- Abubakar H, Sabri SRM, Abdu Masanawa S and Yusuf S (2020b) Modified election algorithm in Hopfield neural network for optimal random k satisfiability representation. *International Journal for Simulation and Multidisciplinary Design Optimization* 11: 16. DOI:10.1051/smdo/2020008.
- Acharya K, Raza W, Dourado C, Velasquez A and Song HH (2023) Neurosymbolic reinforcement learning and planning: A survey. *IEEE Transactions on Artificial Intelligence* 5(5): 1939–1953. URL <https://ieeexplore.ieee.org/stamp/>

- stamp.jsp?arnumber=10238788.
- Ahmed K, Wang E, Chang KW and Van den Broeck G (2022) Neuro-symbolic entropy regularization. In: *Proceedings of the 38th Conference on Uncertainty in Artificial Intelligence (UAI), Proceedings of Machine Learning Research*, volume 180. PMLR, pp. 43–53. URL <https://proceedings.mlr.press/v180/ahmed22a.html>.
- Ahsan F, Maiti S, Chen Z, Kaiser J, Nandi A, Srivatsav M, Schemmel J, Andreou AG, Eshraghian J, Thakur CS and Chakrabartty S (2026) Higher-order neuromorphic Ising machines—autoencoders and Fowler-Nordheim annealers are all you need for scalability. *Nature Communications* 17: 5293. DOI:10.1038/s41467-026-71937-4. URL <https://www.nature.com/articles/s41467-026-71937-4>.
- Akter MS, Sultan MF and Cuzzocrea A (2025) Neuro-symbolic methods in natural language processing: A review. In: *Proceedings of the 14th International Conference on Data Science, Technology and Applications (DATA 2025)*, pages 274–282. URL <https://www.scitepress.org/Papers/2025/134531/134531.pdf>.
- Al-Odhari AM (2021) Features of propositional logic. *Pure Mathematical Sciences* 10(1): 35–44. DOI:10.12988/pms.2021.91275. URL <https://www.m-hikari.com/pms/pms-2021/pms-1-2021/p/alodhariPMS1-2021.pdf>.
- Alberti MA, Bertoni A, Campadelli P, Grossi G and Posenato R (1997) A Neural Algorithm for MAX-2SAT: Performance Analysis and Circuit Implementation. *Neural Networks* 10(3): 555–560. DOI:10.1016/S0893-6080(96)00065-2. URL [https://doi.org/10.1016/S0893-6080\(96\)00065-2](https://doi.org/10.1016/S0893-6080(96)00065-2).
- Ali GA, Abubakar H, Alzaeemi SAS, Almawgani AHM, Sulaiman A and Tay KG (2023) Artificial dragonfly algorithm in the Hopfield neural network for optimal Exact Boolean k satisfiability representation. *PLOS ONE* 18(9). DOI:10.1371/journal.pone.0286874. URL <https://doi.org/10.1371/journal.pone.0286874>.
- Alshiekh M, Bloem R, Ehlers R, Könighofer B, Niekum S and Topcu U (2018) Safe reinforcement learning via shielding. In: *Proceedings of the AAAI conference on artificial intelligence*, volume 32. URL <https://ojs.aaai.org/index.php/AAAI/article/view/11797/11656>.
- Alway A, Kasihmuddin MSM, Mansor MA, Zamri NE, Guo Y, Jamaludin SZM and Kassim AM (2026) Multi-objective optimization via hybrid exhaustive search with intelligent mutation for major 2 satisfiability in Discrete Hopfield Neural Networks. *AIMS Mathematics* 11(5): 15074–15119. DOI:10.3934/math.2026621. URL <https://doi.org/10.3934/math.2026621>.
- Alway A, Zamri NE, Karim SA, Mansor MA, Mohd Kasihmuddin MS and Mohammed Bazuhair M (2022) Major 2 Satisfiability Logic in Discrete Hopfield Neural Network. *International Journal of Computer Mathematics* 99(5): 924–948. DOI:10.1080/00207160.2021.1939870. URL <https://doi.org/10.1080/00207160.2021.1939870>.

- Alzaeemi SA and Sathasivam S (2018) Linear kernel Hopfield neural network approach in horn clause programming. In: *Proceeding of the 25th National Symposium on Mathematical Sciences (SKSM25): Mathematical Sciences as the Core of Intellectual Excellence, AIP Conference Proceedings*, volume 1974. AIP Publishing, p. 020107. DOI:10.1063/1.5041638. URL <https://doi.org/10.1063/1.5041638>.
- Alzaeemi SA, Sathasivam S, Velavan M and Mamat Mb (2020) Agent based Modeling for Activation Function in Enhancement Logic Programming in Hopfield Neural Network. *International Journal of Engineering and Advanced Technology* 9(4): 1872–1879. DOI:10.35940/ijeat.D7711.049420. URL <https://doi.org/10.35940/ijeat.D7711.049420>.
- Amato G, Balestra N, Maggesi M, Parton M et al. (2025) Recurrent neural networks for guiding proof search in propositional logic. In: *Proceedings of the 26th Italian Conference on Theoretical Computer Science*, volume 4039. CEUR-WS, pp. 113–126. URL <https://ceur-ws.org/Vol-4039/paper18.pdf>.
- Amizadeh S, Matushevych S and Weimer M (2018) Learning to solve circuit-sat: An unsupervised differentiable approach. In: *International conference on learning representations*. URL <https://openreview.net/pdf?id=BJxgz2R9t7>.
- Amizadeh S, Matushevych S and Weimer M (2019) PDP: A General Neural Framework for Learning Constraint Satisfaction Solvers. URL <https://arxiv.org/abs/1903.01969>. **Note:** Preprint; acceptance or subsequent publication has not been verified. Consulted arXiv:1903.01969v1; first posted 5 March 2019.
- Andrews R, Diederich J and Tickle AB (1995) Survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge-Based Systems* 8(6): 373–389. DOI:[https://doi.org/10.1016/0950-7051\(96\)81920-4](https://doi.org/10.1016/0950-7051(96)81920-4). URL <https://www.sciencedirect.com/science/article/pii/0950705196819204>.
- Audemard G and Simon L (2018) On the glucose SAT solver. *International Journal on Artificial Intelligence Tools* 27(01): 1840001. DOI:10.1142/S0218213018400018. URL <https://doi.org/10.1142/S0218213018400018>.
- Avigad J (2022) *Semantics of Propositional Logic*. Cambridge University Press, pp. 59–82. DOI:<https://doi.org/10.1017/9781108778756.004>.
- Awolesi AO (2025) Neuro-symbolic generative AI for explainable reasoning. *International Journal of Science and Research Archive* 16(01): 121–125. DOI: 10.30574/ijjsra.2025.16.1.2019. URL <https://doi.org/10.30574/ijjsra.2025.16.1.2019>.
- Azar KZ, Kamali HM, Homayoun H and Sasan A (2020) NNgSAT. In: *Proceedings of the 39th International Conference on Computer-Aided Design*. ACM, pp. 1–9. DOI: 10.1145/3400302.3415669. URL <https://doi.org/10.1145/3400302.3415669>.
- Azizan FL and Sathasivam S (2023) Randomised Alpha-Cut Fuzzy Logic Hybrid Model in Solving 3-Satisfiability Hopfield Neural Network. *Malaysian Journal of Fundamental and Applied Sciences* 19(1): 43–55. DOI:10.11113/mjfas.v19n1.

2697. URL <https://doi.org/10.11113/mjfas.v19n1.2697>.
- Azizan FL, Sathasivam S, Ali MKM, Roslan N and Feng C (2023a) Hybridised Network of Fuzzy Logic and a Genetic Algorithm in Solving 3-Satisfiability Hopfield Neural Networks. *Axioms* 12(3): 250. DOI:10.3390/axioms12030250. URL <https://doi.org/10.3390/axioms12030250>.
- Azizan FL, Sathasivam S and Majahar Ali MK (2023b) Hybridised Intelligent Dynamic Model of 3-Satisfiability Fuzzy Logic Hopfield Neural Network. *Pertanika Journal of Science and Technology* 31(4): 1695–1716. URL <https://api.semanticscholar.org/CorpusID:258901986>.
- Bachmair L and Ganzinger H (2001) Resolution theorem proving. In: Robinson A and Voronkov A (eds.) *Handbook of Automated Reasoning*, volume 1, chapter 2. Elsevier, pp. 19–99. URL <https://lawrencecpaulson.github.io/papers/bachmair-hbar-resolution.pdf>.
- Bader S and Hitzler P (2005) Dimensions of neural-symbolic integration—a structured survey 1: 167–194. URL <https://arxiv.org/abs/cs/0511042>. Published chapter in an edited volume; also available as arXiv:cs/0511042. Peer-review status not independently verified.
- Balendran A, Beji C, Bouvier F, Khalifa O, Evgeniou T, Ravaud P and Porcher R (2025) A scoping review of robustness concepts for machine learning in healthcare. *npj Digital Medicine* 8: 38. DOI:10.1038/s41746-024-01420-1. URL <https://doi.org/10.1038/s41746-024-01420-1>.
- Barwise J (1977) An introduction to first-order logic. In: *Studies in Logic and the Foundations of Mathematics*, volume 90. Elsevier, pp. 5–46. URL <https://www.sciencedirect.com/science/chapter/bookseries/pii/S0049237X08710978>.
- Bazuhair MM, Jamaludin SZM, Zamri NE, Kasihmuddin MSM, Mansor MA, Alway A and Karim SA (2021) Novel Hopfield Neural Network Model with Election Algorithm for Random 3 Satisfiability. *Processes* 9(8): 1292. DOI:10.3390/pr9081292. URL <https://doi.org/10.3390/pr9081292>.
- Belle V (2017) Weighted model counting with function symbols. In: *Conference on Uncertainty in Artificial Intelligence*.
- Berlot-Attwell I (2021) Neuro-symbolic VQA: A review from the perspective of AGI desiderata DOI:10.48550/arXiv.2104.06365. URL <https://arxiv.org/abs/2104.06365>. **Note:** arXiv preprint. Subsequent peer-reviewed publication or acceptance could not be verified as of 12 October 2026.
- Besold TR, d’Avila Garcez A, Bader S, Bowman H, Domingos P, Hitzler P, Kühnberger K, Lamb LC, Lima PMV, de Penning L, Pinkas G, Poon H and Zaverucha G (2021) Neural-symbolic learning and reasoning: A survey and interpretation. In: *Neuro-Symbolic Artificial Intelligence: The State of the Art, Frontiers in Artificial Intelligence and Applications*, volume 342. IOS Press, pp. 1–51. URL <https://ebooks.iospress.nl/volumearticle/58841>. Arxiv: <https://arxiv.org/pdf/1711.03902>.

- Bhuyan BP, Ramdane-Cherif A, Tomar R and Singh TP (2024) Neuro-symbolic artificial intelligence: a survey. *Neural Computing & Applications* 36(21): 12809–12844. DOI:10.1007/s00521-024-09960-z. URL <https://doi.org/10.1007/s00521-024-09960-z>.
- Bin Mohd Kasihmuddin MS, Bin Mansor MA and Sathasivam S (2016) Genetic Algorithm for Restricted Maximum k-Satisfiability in the Hopfield Network. *International Journal of Interactive Multimedia and Artificial Intelligence* 4(2): 52–60. DOI:10.9781/ijimai.2016.429. URL <https://doi.org/10.9781/ijimai.2016.429>.
- Bizzarri A, Yu CEJ, Jalaian B, Riguzzi F and Bastian ND (2025) Neurosymbolic AI for network intrusion detection systems: A survey. *Journal of information security and applications* 94: 104205. URL <https://www.sciencedirect.com/science/article/pii/S221421262500242X>.
- Bonet ML, Levy J and Manyà F (2007) Resolution for Max-SAT. *Artificial Intelligence* 171(8): 606–618. DOI:<https://doi.org/10.1016/j.artint.2007.03.001>. URL <https://www.sciencedirect.com/science/article/pii/S0004370207000422>.
- Bouneffouf D and Aggarwal CC (2022) Survey on applications of neurosymbolic artificial intelligence DOI:10.48550/arXiv.2209.12618. URL <https://arxiv.org/abs/2209.12618>. **Note:** arXiv preprint. Subsequent peer-reviewed publication or acceptance could not be verified as of 12 October 2026.
- Bünz B and Lamm M (2017) Graph Neural Networks and Boolean Satisfiability. URL <https://arxiv.org/abs/1702.03592>. **Note:** Preprint; acceptance or subsequent publication has not been verified. Year is the first arXiv posting (2017); an earlier undated course-hosted copy exists, so the exact first public year is not established.
- Cardillo FA, Khyari H and Straccia U (2025) MILP-SAT-GNN: Yet Another Neural SAT Solver. URL <https://arxiv.org/abs/2507.01825>. **Note:** Research preprint; acceptance or subsequent publication has not been verified. Consulted arXiv:2507.01825v1, first posted 2 July 2025.
- Chang W, Guo M and Luo J (2024) Predicting the satisfiability of Boolean formulas by incorporating gated recurrent unit (GRU) in the Transformer framework. *PeerJ Computer Science* 10: e2169. DOI:10.7717/peerj-cs.2169. URL <https://doi.org/10.7717/peerj-cs.2169>.
- Chang W and Liu W (2025) SAT-GATv2: A Dynamic Attention-Based Graph Neural Network for Solving Boolean Satisfiability Problem. *Electronics* 14(3): 423. DOI:10.3390/electronics14030423. URL <https://doi.org/10.3390/electronics14030423>.
- Chang W, Zhang H and Luo J (2022) Predicting Propositional Satisfiability Based on Graph Attention Networks. *International Journal of Computational Intelligence Systems* 15(1): 84. DOI:10.1007/s44196-022-00139-9. URL <https://doi.org/10.1007/s44196-022-00139-9>.

- Chang Y, Mohd Kasihmuddin MS, Aqlili Ruzai WN, Guo Y and Chen J (2025) Weighted C-type random 2 satisfiability in discrete hopfield neural network. *Engineering Applications of Artificial Intelligence* 160: 111760. DOI:10.1016/j.engappai.2025.111760. URL <https://doi.org/10.1016/j.engappai.2025.111760>.
- Chanin D and Hunter A (2023) Neuro-symbolic commonsense social reasoning. URL <https://arxiv.org/abs/2303.08264>. Cited as an arXiv manuscript; acceptance or publication in a peer-reviewed venue has not been verified.
- Chavira M and Darwiche A (2008) On probabilistic inference by weighted model counting. *Artificial Intelligence* 172(6): 772–799. DOI:<https://doi.org/10.1016/j.artint.2007.11.002>. URL <https://www.sciencedirect.com/science/article/pii/S0004370207001889>.
- Chen H, Li Y, Shi S, Liu S, Zhu H and Zhang Y (2022) Graph collaborative reasoning. In: *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*. pp. 75–84.
- Chen J, Gao Y, Kasihmuddin MSM, Zheng C, Romli NA, Mansor MA, Zamri NE and When C (2024) MTS-PRO2SAT: Hybrid Mutation Tabu Search Algorithm in Optimizing Probabilistic 2 Satisfiability in Discrete Hopfield Neural Network. *Mathematics* 12(5): 721. DOI:10.3390/math12050721. URL <https://doi.org/10.3390/math12050721>.
- Chen MK (2025) A comparative study of neurosymbolic AI approaches to interpretable logical reasoning. In: Gilpin H, Giunchiglia L, Hitzler P and van Krieken E (eds.) *Proceedings of The 19th International Conference on Neurosymbolic Learning and Reasoning, Proceedings of Machine Learning Research*, volume 284. PMLR, pp. 447–462. URL <https://proceedings.mlr.press/v284/chen25b.html>.
- Chen Q, Tan S, Gao S and Lü J (2025) HyperSAT: Unsupervised hypergraph neural networks for weighted MaxSAT problems. DOI:10.48550/arXiv.2504.11885. URL <https://arxiv.org/abs/2504.11885>. **Note:** Preprint subsequently published in the peer-reviewed IEEE Transactions on Neural Networks and Learning Systems, online ahead of print, 2 July 2026, doi:10.1109/TNNLS.2026.3703557.
- Chen X and Sathasivam S (2025) A hybrid firefly algorithm for synaptic weight optimization in discrete Hopfield neural networks applied to random 3-satisfiability problems. *AIMS Mathematics* 10(6): 14840–14892. DOI:10.3934/math.2025667. URL <https://doi.org/10.3934/math.2025667>.
- Chen X and Sathasivam S (2026) An Enhanced Discrete Hopfield Neural Network Framework for Random Boolean Satisfiability Optimization via K-Medoids Clustering and Evolutionary Learning. *Ain Shams Engineering Journal* 17(6): 104160. DOI:10.1016/j.asej.2026.104160. URL <https://doi.org/10.1016/j.asej.2026.104160>.
- Cheng K, Ahmed NK, Rossi RA, Willke T and Sun Y (2025) Neural-symbolic methods for knowledge graph reasoning: A survey. *ACM Transactions on Knowledge Discovery from Data* 18(9): 1–44. URL <https://dl.acm.org/doi/pdf/10.1145/3686806>.

- Colelough BC and Regli W (2025) Neuro-symbolic AI in 2024: A systematic review. *arXiv preprint arXiv:2501.05435* URL <https://arxiv.org/pdf/2501.05435>. Cited as an arXiv preprint; publication in a peer-reviewed venue has not been verified.
- Cormen TH, Leiserson CE, Rivest RL and Stein C (2009) *Introduction to Algorithms*. 3rd edition. The MIT Press. ISBN 9780262033848.
- Dalrymple Dd, Skalse J, Bengio Y, Russell S, Tegmark M, Seshia S, Omohundro S, Szegedy C, Goldhaber B, Ammann N, Abate A, Halpern J, Barrett C, Zhao D, Tan ZX, Wing J and Tenenbaum J (2024) Towards guaranteed safe AI: A framework for ensuring robust and reliable AI systems. URL <https://arxiv.org/abs/2405.06624>. ArXiv preprint. No subsequent peer-reviewed publication or acceptance could be verified as of 12 October 2026.
- d’Anjou A, Graña M, Torrealdea FJ and Hernández MC (1993) Solving satisfiability via Boltzmann machines. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 15(5): 514–521. DOI:10.1109/34.211473. URL <https://doi.org/10.1109/34.211473>.
- d’Avila Garcez AS, Broda K and Gabbay DM (1999) Symbolic knowledge extraction from trained neural networks: The cilp system. *Applied Intelligence* 9(1): 13–33. DOI:10.1023/A:1008325906203. URL <https://www.staff.city.ac.uk/~aag/papers/cilp.pdf>.
- d’Avila Garcez AS, Broda K and Gabbay DM (2012) Neural-symbolic learning systems - foundations and applications. In: *Perspectives in Neural Computing*. URL <https://api.semanticscholar.org/CorpusID:60656463>.
- de Moura L, Kong S, Avigad J, van Doorn F and von Raumer J (2015) The lean theorem prover (system description). In: *Automated Deduction – CADE-25, Lecture Notes in Computer Science*, volume 9195. Springer, pp. 378–388. DOI: 10.1007/978-3-319-21401-6_26.
- Delacour C, Haverkort B, Sabo F, Azemard N and Todri-Sanial A (2025) Lagrange oscillatory neural networks for constraint satisfaction and optimization. *Neuromorphic Computing and Engineering* 5(4): 044004. DOI:10.1088/2634-4386/ae0eab. URL <https://doi.org/10.1088/2634-4386/ae0eab>. **Note:** Peer-reviewed journal article. Published journal version consulted; the publisher PDF states publication on 10 October 2025.
- Delong LN, Mir RF and Fleuriot JD (2025) Neurosymbolic AI for reasoning over knowledge graphs: A survey. *IEEE Trans. Neural Networks Learn. Syst.* 36(5): 7822–7842. URL <https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=10603423>.
- Delvecchio GP, Molfetta L and Moro G (2025) Neuro-symbolic artificial intelligence: A task-directed survey in the black-box models era. In: Kwok J (ed.) *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI-25*. International Joint Conferences on Artificial Intelligence Organization, pp.

- 10418–10426. DOI:10.24963/ijcai.2025/1157. URL <https://doi.org/10.24963/ijcai.2025/1157>.
- Enderton HB (2001) *A mathematical introduction to logic*. Elsevier. URL <https://shop.elsevier.com/books/a-mathematical-introduction-to-logic/enderton/978-0-08-049646-7>.
- Erata F, Kouteili S, Typaldos T, Antonopoulos T, Jones RB, Cook B and Piskac R (2026) Learning How to Cube. URL <https://arxiv.org/abs/2605.16632>. **Note:** Accepted at NeurIPS 2026, supported by the official conference listing and author announcement; proceedings publication not verified as of 8 October 2026. Consulted arXiv:2605.16632v1, first posted 15 May 2026.
- Ertel W, Schumann JMP and Suttner CB (1989) Learning Heuristics for a Theorem Prover using Back Propagation. In: *Informatik-Fachberichte*. Springer Berlin Heidelberg, pp. 87–95. DOI:10.1007/978-3-642-74688-8_10. URL https://doi.org/10.1007/978-3-642-74688-8_10.
- Evans R and Grefenstette E (2018) Learning explanatory rules from noisy data. *Journal of Artificial Intelligence Research* 61: 1–64. URL <https://www.jair.org/index.php/jair/article/download/11172/26376/>.
- Evans R, Saxton D, Amos D, Kohli P and Grefenstette E (2018) Can Neural Networks Understand Logical Entailment? In: *International Conference on Learning Representations*. URL <https://openreview.net/pdf?id=SkZxCK-0Z>.
- Fan Y and Zhang Y (2022) Neural logic analogy learning. In: *Proceedings of the ICLR 2022 Workshop on PAIR² Struct: Privacy, Accountability, Interpretability, Robustness, Reasoning on Structured Data*. URL <https://openreview.net/forum?id=rGe-qsNrx5>. **Note:** Workshop paper, also available as arXiv:2202.02436. The paper-specific peer-review decision could not be independently verified; no subsequent journal or main-conference publication was identified as of 12 October 2026.
- Farooque M, Walter M and Falk A (2024) NeuroDual: A Hybrid SAT Solver Combining Graph Attention Networks with Algorithmic Techniques. In: *2024 International Symposium on Networks, Computers and Communications (ISNCC)*. IEEE, pp. 1–4. DOI:10.1109/ISNCC62547.2024.10759050. URL <https://doi.org/10.1109/ISNCC62547.2024.10759050>.
- Feng C and Sathasivam S (2024) A novel processor for dynamic evolution of constrained SAT problems: The dynamic evolution variant of the discrete Hopfield neural network satisfiability model. *Journal of King Saud University - Computer and Information Sciences* 36(1): 101927. DOI:10.1016/j.jksuci.2024.101927. URL <https://doi.org/10.1016/j.jksuci.2024.101927>.
- Feng C, Sathasivam S, Roslan N and Velavan M (2024) 2-SAT discrete Hopfield neural networks optimization via Crow search and fuzzy dynamical clustering approach. *AIMS Mathematics* 9(4): 9232–9266. DOI:10.3934/math.2024450. URL <https://doi.org/10.3934/math.2024450>.

- Fitas R (2025) Neuro-symbolic AI for advanced signal and image processing: A review of recent trends and future directions. *IEEE Access* URL <https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=11124915>.
- Flint A and Blaschko MB (2012) Perceptron learning of SAT. In: *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., pp. 2771–2779. URL <https://proceedings.neurips.cc/paper/2012/hash/fb60d411a5c5b72b2e7d3527cfc84fd0-Abstract.html>.
- França MV, Zaverucha G and Garcez ASd (2014) Fast relational learning using bottom clause propositionalization with artificial neural networks. *Machine learning* 94(1): 81–104. URL <https://link.springer.com/content/pdf/10.1007/s10994-013-5392-1.pdf>.
- Freivalds K and Kozlovics S (2022) Denoising Diffusion for Sampling SAT Solutions. In: *NeurIPS 2022 Workshop on Score-Based Methods*. URL <https://openreview.net/forum?id=xrHvUr7a6iW>. **Note:** Verified accepted workshop paper; the official call specifies double-blind review. Not a NeurIPS main-track paper. Consulted arXiv:2212.00121v1, first posted 30 November 2022. No journal or main-conference publication verified.
- Funahashi KI (1989) On the approximate realization of continuous mappings by neural networks. *Neural networks* 2(3): 183–192. URL <https://math.bu.edu/people/mkon/MA751/FunahashiTheorem.pdf>.
- Gao Y, Guo Y, Romli NA, Kasihmuddin MSM, Chen W, Mansor MA and Chen J (2022) GRAN3SAT: Creating Flexible Higher-Order Logic Satisfiability in the Discrete Hopfield Neural Network. *Mathematics* 10(11): 1899. DOI:10.3390/math10111899. URL <https://doi.org/10.3390/math10111899>.
- Gao Y, Kasihmuddin MSM, Chen J, Zheng C, Romli NA, Mansor MA and Zamri NE (2024) Binary ant colony optimization algorithm in learning random satisfiability logic for discrete Hopfield neural network. *Applied Soft Computing* 166: 112192. DOI:10.1016/j.asoc.2024.112192. URL <https://doi.org/10.1016/j.asoc.2024.112192>.
- Garcez Ad, Gori M, Lamb LC, Serafini L, Spranger M and Tran SN (2019) Neural-symbolic computing: An effective methodology for principled integration of machine learning and reasoning. *Journal of Applied Logics* 6(4): 611–631. URL <https://arxiv.org/abs/1905.06088>. Published in the peer-reviewed Journal of Applied Logics, vol. 6, no. 4 (2019); a version of the article is also available as arXiv:1905.06088.
- Garcez ASd, Gabbay DM, Ray O and Woods J (2007) Abductive reasoning in neural-symbolic systems. *Topoi* 26(1): 37–49. URL <https://link.springer.com/content/pdf/10.1007/s11245-006-9005-5.pdf>.
- Garcez ASd and Zaverucha G (1999) The connectionist inductive learning and logic programming system. *Applied Intelligence* 11(1): 59–77. URL <https://link.springer.com/content/pdf/10.1023/A:1008328630915.pdf>.

- Garey MR and Johnson DS (1979) *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: W. H. Freeman and Company. URL <https://perso.limos.fr/~palafour/PAPERS/PDF/Garey-Johnson79.pdf>.
- Gauthier T (2020) Tree neural networks in HOL4. In: *Intelligent Computer Mathematics, Lecture Notes in Computer Science*, volume 12236. Cham: Springer, pp. 278–283. DOI:10.1007/978-3-030-53518-6_18. URL https://doi.org/10.1007/978-3-030-53518-6_18.
- Ghanem MA (2024) NeuRes: A neural resolution prover of unsatisfiability URL https://publications.cispa.de/articles/thesis/NeuRes_A_Neural_Resolution_Prover_of_Unsatisfiability/27118017/1/files/49444704.pdf.
- Gibaut W, Pereira L, Grassiotto F, Osorio A, Gadioli E, Munoz A, Gomes S and Santos Cd (2023) Neurosymbolic AI and its taxonomy: a survey. *arXiv preprint arXiv:2305.08876* DOI:10.48550/arXiv.2305.08876. Cited as an arXiv preprint; acceptance or publication in a peer-reviewed venue has not been verified.
- Gordeev L and Haeusler EH (2019) Proof compression and np versus pspace. *Studia Logica* 107(1): 53–83. URL <https://link.springer.com/article/10.1007/s11225-017-9773-5>.
- Gorré R (2006) An introduction to classical propositional logic: Syntax, semantics, sequents. In: Lauer H (ed.) *Foundations of the Pure Sciences, in History and Philosophy of Science for Science Students Committed to Africa*, chapter 49. Ghana University Press, Lagon, Ghana, pp. 3–45. URL <https://www.cs.brandeis.edu/~cs112/2006-cs112/docs/intrologic1.pdf>.
- Guo Y, Kasihmuddin MSM, Gao Y, Mansor MA, Wahab HA, Zamri NE and Chen J (2022) YRAN2SAT: A novel flexible random satisfiability logical rule in discrete hopfield neural network. *Advances in Engineering Software* 171: 103169. DOI:10.1016/j.advengsoft.2022.103169. URL <https://doi.org/10.1016/j.advengsoft.2022.103169>.
- Guo Y, Zamri NE, Kasihmuddin MSM, Alway A, Mansor MA, Li J and Zhang Q (2024) Dual optimization approach in discrete Hopfield neural network. *Applied Soft Computing* 164: 111929. DOI:10.1016/j.asoc.2024.111929. URL <https://doi.org/10.1016/j.asoc.2024.111929>.
- Gurney K (1997) *An Introduction to Neural Networks*. London and New York: UCL Press. URL <https://www.routledge.com/An-Introduction-to-Neural-Networks/Gurney/p/book/9781315273570>.
- Hamadneh N, Sathasivam S and Ong HC (2014a) Computing Single Step Operators of Logic Programming in Radial Basis Function Neural Networks. In: *AIP Conference Proceedings*, volume 1605. AIP Publishing, pp. 458–463. DOI:10.1063/1.4887632. URL <https://doi.org/10.1063/1.4887632>.
- Hamadneh N, Sathasivam S, Tilahun SL and Ong HC (2014b) Satisfiability of logic programming based on radial basis function neural networks. In: *AIP Conference Proceedings*, volume 1605. AIP Publishing LLC, pp. 547–550. DOI:10.1063/1.

4887647. URL <https://doi.org/10.1063/1.4887647>.

Hamilton K, Nayak A, Božić B and Longo L (2022) Is neuro-symbolic AI meeting its promises in natural language processing? a structured review. *Semantic Web – Interoperability, Usability, Applicability* URL <https://www.semantic-web-journal.net/system/files/swj3228.pdf>.

Han JM (2020) Enhancing SAT solvers with glue variable predictions. URL <https://arxiv.org/abs/2007.02559>. **Note:** Preprint; acceptance or subsequent publication has not been verified. Consulted arXiv:2007.02559v1, first posted 6 July 2020.

Harrison J (2009) *Handbook of Practical Logic and Automated Reasoning*. Cambridge University Press. ISBN 9780521899574. URL https://kwarc.info/teaching/sWuV/harrison_handbook-of-practical-logic.pdf.

Hashemi N, Hoxha B, Yamaguchi T, Prokhorov D, Fainekos G and Deshmukh J (2023) A neurosymbolic approach to the verification of temporal logic properties of learning-enabled control systems. In: *Proceedings of the 14th ACM/IEEE International Conference on Cyber-Physical Systems*. ACM, pp. 98–109. DOI: 10.1145/3576841.3585928.

Hatzilygeroudis I and Prentzas J (2000) Neurules: improving the performance of symbolic rules. *International Journal on Artificial Intelligence Tools* 9(01): 113–130. URL <https://www.worldscientific.com/doi/abs/10.1142/S0218213000000094>.

Heule MJ, Hunt WA and Wetzler N (2013) Trimming while checking clausal proofs. In: *2013 Formal Methods in Computer-Aided Design*. pp. 181–188. DOI:10.1109/FMCAD.2013.6679408.

Heule MJH, Kiesel B, Seidl M and Biere A (2017) Pruning through satisfaction. In: *Haifa Verification Conference (HVC)*. Springer, pp. 179–194. DOI: 10.1007/978-3-319-67205-8_13. URL <https://www.cs.utexas.edu/~marijn/publications/PRune.pdf>.

Hitzler P, Hölldobler S and Seda AK (2004) Logic programs and connectionist networks. *Journal of Applied Logic* 2(3): 245–272. DOI:<https://doi.org/10.1016/j.jal.2004.03.002>. URL <https://www.sciencedirect.com/science/article/pii/S1570868304000151>.

Hitzler P, Krötzsch M, Parsia B, Patel-Schneider PF and Rudolph S (2012) OWL 2 Web Ontology Language Primer (second edition). World Wide Web Consortium, W3C Recommendation, 11 December 2012. URL <https://www.w3.org/TR/2012/REC-owl2-primer-20121211/>. Listed contributors are editors.

Hou S (2026) GMTSAT: A Neural SAT Solver Based on Multi-Slot Global Updates and Gated Message Passing. *International Journal of Computer Science and Information Technology* 8(2): 70–91. DOI:10.62051/ijcsit.v8n2.08. URL <https://ijocsit.org/articles/8-2-gmstsat-a-neural-sat-solver-based-on-multi-slot-global-update/>.

- Hu Y, Wang W, Khurshid S, McMillan KL and Tiwari M (2023) Fixing Privilege Escalations in Cloud Access Control with MaxSAT and Graph Neural Networks. In: *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, pp. 104–115. DOI:10.1109/ASE56229.2023.00167. URL <https://doi.org/10.1109/ASE56229.2023.00167>.
- Huang J, Li Z, Naik M and Lim SN (2025) Laser: A neuro-symbolic framework for learning spatial-temporal scene graphs with weak supervision. URL <https://arxiv.org/abs/2304.07647>. Cited as an arXiv manuscript; acceptance or publication in a peer-reviewed venue has not been verified.
- Hůla J, Mojžíšek D and Janota M (2024) Understanding GNNs for Boolean Satisfiability through Approximation Algorithms. In: *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*. Association for Computing Machinery, pp. 953–961. DOI:10.1145/3627673.3679813. URL <https://doi.org/10.1145/3627673.3679813>.
- Jaakkola R, Janhunen T, Kuusisto A, Rankooh MF and Vilander M (2025) Explainability via short formulas: the case of propositional logic with implementation. *Journal of Artificial Intelligence Research* 83. URL <https://www.jair.org/index.php/jair/article/view/17422>.
- Jarmužek T (2020) *Tableau Methods for Propositional Logic and Term Logic, Studies in Philosophy, History of Ideas and Modern Societies*, volume 20. Berlin, Bern, Bruxelles, New York, Oxford, Warszawa, Wien: Peter Lang GmbH, Internationaler Verlag der Wissenschaften. ISBN 9783631833728. DOI:10.3726/b18008. URL <https://doi.org/10.3726/b18008>.
- Jaszczur S, Łuszczuk M and Michalewski H (2019) Neural heuristics for SAT solving. In: *Representation Learning on Graphs and Manifolds Workshop at ICLR 2019*. URL <https://rlgm.github.io/papers/32.pdf>. **Note:** Workshop paper and acceptance verified from the official ICLR 2019 RLGM workshop list; peer-review process not independently verified. Not an ICLR main-track paper. Consulted arXiv:2005.13406v1, first posted 27 May 2020; the earlier workshop version is from 2019.
- Jiang X, Kasihmuddin MSM, Guo Y, Gao Y, Mansor MA, Zamri NE and Marsani MF (2024) J-type random 2,3 satisfiability: a higher-order logical rule in discrete hopfield neural network. *Evolutionary Intelligence* 17(5-6): 3317–3336. DOI:10.1007/s12065-024-00936-5. URL <https://doi.org/10.1007/s12065-024-00936-5>.
- Johnson JL (1989) A neural network approach to the 3-satisfiability problem. *Journal of Parallel and Distributed Computing* 6(2): 435–449. DOI:10.1016/0743-7315(89)90068-3. URL [https://doi.org/10.1016/0743-7315\(89\)90068-3](https://doi.org/10.1016/0743-7315(89)90068-3).
- Jönsson H and Söderberg B (2001) An Information-Based Neural Approach to Constraint Satisfaction. *Neural Computation* 13(8): 1827–1838. DOI:10.1162/08997660152469369. URL <https://doi.org/10.1162/08997660152469369>.

Note: Verified journal publication in Neural Computation (2001). Consulted arXiv:cond-mat/0105319v1, first posted 16 May 2001; counted once using the journal year.

- Karim SA, Kasihmuddin MSM, Mansor MA, Alway A and Zamri NE (2022) A comparative performance analysis between 2-satisfiability and random 3-satisfiability in Discrete Hopfield Neural Network. In: *AIP Conference Proceedings*, volume 2465. AIP Publishing, p. 020006. DOI:10.1063/5.0078616. URL <https://doi.org/10.1063/5.0078616>.
- Karim SA, Zamri NE, Alway A, Mohd Kasihmuddin MS, Md Ismail AI, Mansor MA and Abu Hassan NF (2021) Random Satisfiability: A Higher-Order Logical Approach in Discrete Hopfield Neural Network. *IEEE Access* 9: 50831–50845. DOI:10.1109/ACCESS.2021.3068998. URL <https://doi.org/10.1109/ACCESS.2021.3068998>.
- Kasihmuddin MSbM and Sathasivam S (2016) Accelerating activation function in higher order logic programming. In: *AIP Conference Proceedings*, volume 1750. p. 030006. DOI:10.1063/1.4954542. URL <https://doi.org/10.1063/1.4954542>.
- Kasihmuddin MSM, Mansor MA, Alzaeemi S, Basir MFM and Sathasivam S (2019) Quality Solution of Logic Programming in Hopfield Neural Network. *Journal of Physics: Conference Series* 1366(1): 012094. DOI:10.1088/1742-6596/1366/1/012094. URL <https://doi.org/10.1088/1742-6596/1366/1/012094>.
- Kasihmuddin MSM, Mansor MA and Sathasivam S (2016a) Artificial Bee Colony in the Hopfield Network for Maximum k-Satisfiability Problem. *Journal of Informatics and Mathematical Sciences* 8(5): 317–334. URL <https://www.rgnpublications.com/journals/index.php/jims/article/view/467>.
- Kasihmuddin MSM, Mansor MA and Sathasivam S (2016b) Bezier Curves Satisfiability Model in Enhanced Hopfield Network. *International Journal of Intelligent Systems and Applications* 8(12): 9–17. DOI:10.5815/ijisa.2016.12.02. URL <https://doi.org/10.5815/ijisa.2016.12.02>.
- Kathirvel V, Mansor MA, Mohd Kasihmuddin MS and Sathasivam S (2019) Hybrid Imperialistic Competitive Algorithm Incorporated with Hopfield Neural Network for Robust 3 Satisfiability Logic Programming. *IAES International Journal of Artificial Intelligence (IJ-AI)* 8(2): 144–155. DOI:10.11591/ijai.v8.i2.pp144-155. URL <https://doi.org/10.11591/ijai.v8.i2.pp144-155>.
- Kautz H (2022) The third AI summer: AAAI robert s. engelmore memorial lecture. *Ai magazine* 43(1): 105–125. DOI:<https://doi.org/10.1002/aaai.12036>. URL <https://ojs.aaai.org/aimagazine/index.php/aimagazine/article/view/19122>.
- Kho LC, Kasihmuddin MS, Mansor MA and Sathasivam S (2022) Propositional satisfiability logic via ant colony optimization in hopfield neural network. *Malaysian Journal of Mathematical Sciences* 16(1). URL <http://119.40.117.179/fullpaper/2022-January-16-1/Kasihmuddin,%20M.%20S.%20M.-37-53.pdf>.
- Kho LC, Kasihmuddin MSM, Mansor MA and Sathasivam S (2019) 2 satisfiability logical rule by using ant colony optimization in Hopfield Neural Network. In:

- AIP Conference Proceedings, volume 2184. AIP Publishing, p. 060009. DOI: 10.1063/1.5136441. URL <https://doi.org/10.1063/1.5136441>.
- Kibria RH (2007) Evolving a Neural Net-Based Decision and Search Heuristic for DPLL SAT Solvers. In: *2007 International Joint Conference on Neural Networks*. IEEE, pp. 765–770. DOI:10.1109/IJCNN.2007.4371054. URL <https://doi.org/10.1109/IJCNN.2007.4371054>.
- Kraetzschmar G, Sablatnög S, Enderle S and Palm G (1998) Application of neurosymbolic integration for environment modelling in mobile robots. In: *International Workshop on Hybrid Neural Systems*. Springer, pp. 387–401. URL https://link.springer.com/chapter/10.1007/10719871_27.
- Kurin V, Godil S, Whiteson S and Catanzaro B (2020) Can Q-Learning with graph networks learn a generalizable branching heuristic for a SAT solver? In: *Advances in Neural Information Processing Systems*, volume 33. pp. 9608–9621. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/6d70cb65d15211726dcce4c0e971e21c-Paper.pdf.
- Lamb LC, Garcez A, Gori M, Prates M, Avelar P and Vardi M (2020) Graph neural networks meet neural-symbolic computing: A survey and perspective. *arXiv preprint arXiv:2003.00330* URL <https://arxiv.org/pdf/2003.00330>. A peer-reviewed version was published in the Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20), Survey Track, pp. 4877–4884, 2020, doi:10.24963/ijcai.2020/679.
- Lammich P (2020) Efficient verified (un) sat certificate checking. *Journal of Automated Reasoning* 64(3): 513–532. URL <https://doi.org/10.1007/s10817-019-09525-z>.
- Langedijk A, Jumelet J and Zuidema W (2025) Propositional logic for probing generalization in neural networks DOI:10.48550/arXiv.2506.08978. URL <https://arxiv.org/abs/2506.08978>. **Note:** arXiv preprint. Subsequent peer-reviewed publication or acceptance could not be verified as of 12 October 2026.
- LeCun Y, Bottou L, Bengio Y and Haffner P (1998) Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11): 2278–2324. DOI: 10.1109/5.726791.
- Li M, Shi Z, Lai Q, Khan S, Cai S and Xu Q (2023) On EDA-Driven Learning for SAT Solving. In: *2023 60th ACM/IEEE Design Automation Conference (DAC)*. IEEE, pp. 1–6. DOI:10.1109/DAC56929.2023.10248001. URL <https://doi.org/10.1109/DAC56929.2023.10248001>.
- Li V, Fu Y, Knappe T, Han K and Zhu K (2025) Automating mathematical proof generation using large language model agents and knowledge graphs. *arXiv preprint abs/2503.11657*. URL <https://arxiv.org/abs/2503.11657>. Cited as an arXiv manuscript; acceptance or publication in a peer-reviewed venue has not been verified.

- Li Z and Si X (2022) NSNet: A general neural probabilistic framework for satisfiability problems. In: *Advances in Neural Information Processing Systems*, volume 35. Curran Associates, Inc., pp. 25573–25585. DOI:10.52202/068431-1854. URL <https://doi.org/10.52202/068431-1854>.
- Li Z, Sun J, Murphy L, Su Q, Li Z, Zhang X, Yang K and Si X (2024) A survey on deep learning for theorem proving. *arXiv preprint arXiv:2404.09939* URL <https://arxiv.org/pdf/2404.09939>. Cited as an arXiv preprint; acceptance or publication in a peer-reviewed venue has not been verified.
- Liang B, Wang Y and Tong C (2025) Ai reasoning in deep learning era: From symbolic AI to neural-symbolic ai. *Mathematics* 13(11): 1707. URL <https://www.mdpi.com/2227-7390/13/11/1707>.
- Liberatore P (2008) Redundancy in logic ii: 2cnf and horn propositional formulae. *Artificial Intelligence* 172(2-3): 265–299. DOI:10.1016/j.artint.2007.06.003. URL <https://www.sciencedirect.com/science/article/pii/S0004370207001105>.
- Lima PMV (2001) A goal-driven neural propositional interpreter. *International Journal of Neural Systems* 11(3): 311–322. DOI:10.1142/S012906570100076X. URL <https://doi.org/10.1142/S012906570100076X>.
- Lima PMV, Morveli-Espinoza MMM, Pereira GC, Ferreira TO and França FMG (2008) Logical reasoning via satisfiability mapped into energy functions. *International Journal of Pattern Recognition and Artificial Intelligence* 22(5): 1031–1043. DOI:10.1142/S0218001408006673. URL <https://doi.org/10.1142/S0218001408006673>.
- Liu C, Liu G, Luo C, Cai S, Lei Z, Zhang W, Chu Y and Zhang G (2025a) Optimizing local search-based partial MaxSAT solving via initial assignment prediction. *Science China Information Sciences* 68(2): 122101. DOI:10.1007/s11432-023-3900-7. URL <https://doi.org/10.1007/s11432-023-3900-7>.
- Liu M, Huang P, Jia F, Zhang F, Sun Y, Cai S, Ma F and Zhang J (2023) Can Graph Neural Networks Learn to Solve the MaxSAT Problem? *Proceedings of the AAAI Conference on Artificial Intelligence* 37(13): 16264–16265. DOI:10.1609/aaai.v37i13.26992. URL <https://ojs.aaai.org/index.php/AAAI/article/view/26992>.
- Liu X, Chang Y and Jiang X (2025b) Weighted random 3-satisfiability in discrete Hopfield neural network. *Mathematical Modeling and Computing* 12(2): 468–480. DOI:10.23939/mmc2025.02.468. URL <https://doi.org/10.23939/mmc2025.02.468>.
- Lymperopoulos P and Liu L (2024) Graph Pruning for Enumeration of Minimal Unsatisfiable Subsets. In: *Proceedings of the 27th International Conference on Artificial Intelligence and Statistics, Proceedings of Machine Learning Research*, volume 238. PMLR, pp. 2647–2655. URL <https://proceedings.mlr.press/v238/lymperopoulos24a.html>.

- Manhaeve R, Dumancic S, Kimmig A, Demeester T and De Raedt L (2018) Deep-problog: Neural probabilistic logic programming. *Advances in neural information processing systems* 31. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/dc5d637ed5e62c36ecb73b654b05ba2a-Paper.pdf.
- Manoharam G, Kassim AM, Abdeen S, Kasihmuddin MSM, Rusdi NA, Romli NA, Zamri NE and Mansor MA (2024) Special major 1, 3 satisfiability logic in discrete Hopfield neural networks. *AIMS Mathematics* 9(5): 12090–12127. DOI:10.3934/math.2024591. URL <https://doi.org/10.3934/math.2024591>.
- Mansor MA, Kasihmuddin MSM and Sathasivam S (2016a) Enhanced Hopfield network for pattern satisfiability optimization. *International Journal of Intelligent Systems and Applications* 8(11): 27–33. DOI:10.5815/ijisa.2016.11.04. URL <https://doi.org/10.5815/ijisa.2016.11.04>.
- Mansor MA, Kasihmuddin MSM and Sathasivam S (2016b) VLSI circuit configuration using satisfiability logic in Hopfield network. *International Journal of Intelligent Systems and Applications* 8(9): 22–29. DOI:10.5815/ijisa.2016.09.03. URL <https://doi.org/10.5815/ijisa.2016.09.03>.
- Mansor MA, Kasihmuddin MSM and Sathasivam S (2017a) Artificial immune system paradigm in the Hopfield network for 3-satisfiability problem. *Pertanika Journal of Science & Technology* 25(4): 1173–1188. URL https://web.archive.org/web/20180520051652id_/http://www.myjurnal.my/filebank/published_article/58029/10.pdf.
- Mansor MA, Kasihmuddin MSM and Sathasivam S (2019) Modified Artificial Immune System Algorithm with Elliot Hopfield Neural Network For 3-Satisfiability Programming. *Journal of Informatics and Mathematical Sciences* 11(1): 81–98. URL <https://www.rgnpublications.com/journals/index.php/jims/article/view/1062>.
- Mansor MA, Kasihmuddin MSM and Sathasivam S (2020) Modified Lion Optimization Algorithm with Discrete Hopfield Neural Network for Higher Order Boolean Satisfiability Programming. *Malaysian Journal of Mathematical Sciences* 14(S): 47–61. URL [https://mjms.upm.edu.my/lihatmakalah.php?kod=2020/December/14\(S\)//47-61](https://mjms.upm.edu.my/lihatmakalah.php?kod=2020/December/14(S)//47-61).
- Mansor MA, Kasihmuddin MSM and Sathasivam S (2021) Grey Wolf Optimization algorithm with Discrete Hopfield Neural Network for 3 Satisfiability analysis. *Journal of Physics: Conference Series* 1821(1). DOI:10.1088/1742-6596/1821/1/012038. URL <https://doi.org/10.1088/1742-6596/1821/1/012038>.
- Mansor MA and Sathasivam S (2016a) Accelerating activation function for 3-satisfiability logic programming. *International Journal of Intelligent Systems and Applications* 8(10): 44–50. DOI:10.5815/ijisa.2016.10.05. URL <https://doi.org/10.5815/ijisa.2016.10.05>.
- Mansor MA and Sathasivam S (2016b) Performance analysis of activation function in higher order logic programming. In: *Advances in Industrial and Applied*

Mathematics: Proceedings of the 23rd Malaysian National Symposium of Mathematical Sciences (SKSM23), AIP Conference Proceedings, volume 1750. AIP Publishing, p. 030007. DOI:10.1063/1.4954543. URL <https://doi.org/10.1063/1.4954543>.

Mansor MA and Sathasivam S (2021) Optimal Performance Evaluation Metrics for Satisfiability Logic Representation in Discrete Hopfield Neural Network. *International Journal of Mathematics and Computer Science* 16(3): 963–976. URL <https://future-in-tech.net/16.3/R-Mansor-Sathasivam.pdf>.

Mansor MAB, Kasihmuddin MSBM and Sathasivam S (2017b) Robust Artificial Immune System in the Hopfield network for Maximum k-Satisfiability. *International Journal of Interactive Multimedia and Artificial Intelligence* 4(4): 63–71. DOI:10.9781/ijimai.2017.448.

Marino R (2021) Learning from Survey Propagation: A Neural Network for MAX-E-3-SAT. *Machine Learning: Science and Technology* 2(3): 035032. DOI:10.1088/2632-2153/ac0496. URL <https://doi.org/10.1088/2632-2153/ac0496>.

Marra G, Dumančić S, Manhaeve R and De Raedt L (2024) From statistical relational to neurosymbolic artificial intelligence: A survey. *Artificial Intelligence* 328: 104062. DOI:<https://doi.org/10.1016/j.artint.2023.104062>. URL <https://www.sciencedirect.com/science/article/pii/S0004370223002084>.

McCulloch WS and Pitts W (1943) A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics* 5(4): 115–133. URL <https://jontalle.web.engr.illinois.edu/uploads/410-NS.F22/McCulloch-Pitts-1943-neural-networks-ocr.pdf>.

McCune W (2005–2010) Prover9 and mace4. URL <https://github.com/ai4reason/Prover9>.

McGuinness L (2021) Prisma 2020 checklist shiny user-friendly interface (app). URL <https://prisma.shinyapps.io/checklist/>. **Note:** User-friendly interface for the PRISMA 2020 checklist. Accessed 12 Nov 2025.

Miles A and Bechhofer S (????) SKOS Vocabulary: Turtle Serialization. Copy hosted in the OpenLink Software RDF Editor repository. URL <https://raw.githubusercontent.com/openlink/rdf-editor/develop/app/data/skos.ttl>. Accessed 23 September 2026.

Miles A and Bechhofer S (2009) SKOS Simple Knowledge Organization System Reference. URL <https://www.w3.org/TR/2009/REC-skos-reference-20090818/>. Edited by Alistair Miles and Sean Bechhofer.

Mohd Kasihmuddin MS, Mansor MA, Md Basir MF and Sathasivam S (2019) Discrete Mutation Hopfield Neural Network in Propositional Satisfiability. *Mathematics* 7(11): 1133. DOI:10.3390/math7111133. URL <https://doi.org/10.3390/math7111133>.

Mohd Kasihmuddin MS, Mansor MA and Sathasivam S (2018) Discrete Hopfield Neural Network in Restricted Maximum k-Satisfiability Logic Programming. *Sains*

- Malaysiana 47(6): 1327–1335. DOI:10.17576/jsm-2018-4706-30. URL <https://doi.org/10.17576/jsm-2018-4706-30>.
- Mojzisek D, Hula J, Li Z, Zhou Z and Janota M (2025) Neural approaches to sat solving: Design choices and interpretability. *arXiv preprint arXiv:2504.01173* URL <https://arxiv.org/pdf/2504.01173> **Note:** Preprint subsequently published in the peer-reviewed International Journal of Approximate Reasoning, volume 190, article 109609, 2026, doi:10.1016/j.ijar.2025.109609.
- Molnár B and Ercsey-Ravasz M (2013) Asymmetric Continuous-Time Neural Networks without Local Traps for Solving Constraint Satisfaction Problems. *PLoS ONE* 8(9): e73400. DOI:10.1371/journal.pone.0073400. URL <https://doi.org/10.1371/journal.pone.0073400>.
- Monfreglio A (1991) Connectionist networks for constraint satisfaction. *Neurocomputing* 3(1): 29–49. DOI:10.1016/0925-2312(91)90018-7. URL [https://doi.org/10.1016/0925-2312\(91\)90018-7](https://doi.org/10.1016/0925-2312(91)90018-7).
- Monfreglio A (1993) Neural networks for constraint satisfaction. *Connection Science* 5(2): 169–187. DOI:10.1080/09540099308915694. URL <https://doi.org/10.1080/09540099308915694>.
- Monfreglio A (1994) Backpropagation networks for logic constraint solving. *Neurocomputing* 6(1): 67–98. DOI:10.1016/0925-2312(94)90035-3. URL [https://doi.org/10.1016/0925-2312\(94\)90035-3](https://doi.org/10.1016/0925-2312(94)90035-3).
- Monfreglio A (1995) Neural networks for finite constraint satisfaction. *Neural Computing & Applications* 3(2): 78–100. DOI:10.1007/BF01421960. URL <https://doi.org/10.1007/BF01421960>.
- Monfreglio A (1999) Neural Networks and Linear Programming for the Satisfiability Problem. *International Journal of Neural Systems* 9(1): 11–25. DOI:10.1142/S0129065799000034. URL <https://doi.org/10.1142/S0129065799000034>.
- Montgomery DC and Runger GC (2014) *Applied Statistics and Probability for Engineers*. 6th edition. Hoboken, NJ, USA: Wiley. See chapter on measures of central tendency for sample mean and median definitions.
- Moriyama S and Inoue K (2025) Graph-Based Attention for Differentiable MaxSAT Solving. In: *Advances in Neural Information Processing Systems 38*. Neural Information Processing Systems Foundation, Inc. (NeurIPS), pp. 152819–152847. DOI:10.52202/085713-4598. URL <https://doi.org/10.52202/085713-4598>.
- Moriyama S, Watanabe K and Inoue K (2023) GNN Based Extraction of Minimal Unsatisfiable Subsets. In: *Inductive Logic Programming, Lecture Notes in Computer Science*, volume 14363. Springer Nature Switzerland, pp. 77–92. DOI:10.1007/978-3-031-49299-0_6. URL https://doi.org/10.1007/978-3-031-49299-0_6.
- Nagamatu M, Akahoshi T, Okumura M, Miyamoto K and Aman H (2000) Controlling attention to clauses for solving SAT by lagrange programming neural network. *Biomedical Soft Computing and Human Sciences* 5(2): 65–71. DOI:10.24466/

ijbschs.5.2_65. URL https://doi.org/10.24466/ijbschs.5.2_65.

Nagamatu M and Yanaru T (1996) On the stability of Lagrange programming neural networks for satisfiability problems of prepositional calculus. *Neurocomputing* 13(2–4): 119–133. DOI:10.1016/0925-2312(95)00087-9. URL [https://doi.org/10.1016/0925-2312\(95\)00087-9](https://doi.org/10.1016/0925-2312(95)00087-9).

Narazaki H and Ralescu A (1992) A connectionist approach for rule-based inference using an improved relaxation method. *IEEE Transactions on Neural Networks* 3(5): 741–751. DOI:10.1109/72.159062. URL <https://doi.org/10.1109/72.159062>.

Nawaz U, ur Rahaman MA and Saeed Z (2025) A review of neuro-symbolic AI integrating reasoning and learning for advanced cognitive systems. *Intelligent Systems with Applications* 26: 200541. DOI:<https://doi.org/10.1016/j.iswa.2025.200541>. URL <https://www.sciencedirect.com/science/article/pii/S2667305325000675>.

Nilsson NJ (1986) Probabilistic logic. *Artificial intelligence* 28(1): 71–87. URL <https://www.sciencedirect.com/science/article/pii/0004370286900317>.

Ozolins E, Freivalds K, Draguns R, Gaile E, Zakovskis R and Kozlovics S (2022) Goal-aware neural SAT solver. In: *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE, pp. 1–8. DOI:10.1109/IJCNN55064.2022.9892733. URL <https://doi.org/10.1109/IJCNN55064.2022.9892733>.

Page MJ, McKenzie JE, Bossuyt PM, Boutron I, Hoffmann TC, Mulrow CD, Shamseer L, Tetzlaff JM, Akl EA, Brennan SE, Chou R, Glanville J, Grimshaw JM, Hróbjartsson A, Lalu MM, Li T, Loder EW, Mayo-Wilson E, McDonald S, McGuinness LA, Stewart LA, Thomas J, Tricco AC, Welch VA, Whiting P and Moher D (2021) The prisma 2020 statement: an updated guideline for reporting systematic reviews. *BMJ* 372. DOI:10.1136/bmj.n71. URL <https://www.bmj.com/content/372/bmj.n71>.

Pan L, Ganesh V, Abernethy J, Esposo C and Lee W (2025) Can Transformers Reason Logically? A Study in SAT Solving. In: *Proceedings of the 42nd International Conference on Machine Learning, Proceedings of Machine Learning Research*, volume 267. PMLR, pp. 47632–47671. URL <https://proceedings.mlr.press/v267/pan25d.html>.

Pelosi D, Cacciagrano D and Piangerelli M (2025) Explainability and interpretability in concept and data drift: A systematic literature review. *Algorithms* 18(7). DOI: 10.3390/a18070443. URL <https://www.mdpi.com/1999-4893/18/7/443>.

Peltonen S and Wattenhofer R (2026) On the Expressive Power of GNNs for Boolean Satisfiability. In: *The Fourteenth International Conference on Learning Representations*. ICLR. URL <https://openreview.net/forum?id=Q01JX3CuDx>.

Pinkas G (1991a) Propositional non-monotonic reasoning and inconsistency in symmetric neural networks. In *IJCAI* 91: 525–530. URL https://openscholarship.wustl.edu/cgi/viewcontent.cgi?article=1622&context=cse_research.

- Pinkas G (1991b) Symmetric neural networks and propositional logic satisfiability. *Neural Computation* 3(2): 282–291. URL <https://doi.org/10.1162/neco.1991.3.2.282>.
- Pinkas G (1995) Reasoning, nonmonotonicity and learning in connectionist networks that capture propositional knowledge. *Artificial Intelligence* 77(2): 203–247. DOI: 10.1016/0004-3702(94)00032-V.
- Plaisted DA (2014) Automated theorem proving. *Wiley Interdisciplinary Reviews: Cognitive Science* 5(2): 115–128. DOI:10.1002/wcs.1269. URL <https://doi.org/10.1002/wcs.1269>.
- Poole DL and Mackworth AK (2010) *Artificial Intelligence: foundations of computational agents*. Cambridge University Press. URL <https://www.cambridge.org/highereducation/books/artificial-intelligence/C113F6CE284AB00F5489EBA5A59B93B7#overview>.
- Rasga J and Sernadas C (2022) Adding abductive reasoning to a propositional logic. *Journal of Logic and Computation* 32(4): 697–728. URL <https://academic.oup.com/logcom/article/32/4/697/6457177>.
- Rasli NFA, Kasihmuddin MSM, Mansor MA, Basir MFM and Sathasivam S (2020) k satisfiability programming by using estimation of distribution algorithm in Hopfield neural network. In: *AIP Conference Proceedings*, volume 2266. AIP Publishing, p. 040007. DOI:10.1063/5.0018263. URL <https://doi.org/10.1063/5.0018263>.
- Renkhoff J, Feng K, Meier-Doernberg M, Velasquez A and Song HH (2024) A survey on verification and validation, testing and evaluations of neurosymbolic artificial intelligence. *IEEE Transactions on Artificial Intelligence* 5(8): 3765–3779. URL <https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=10385139>.
- Riegel R, Gray A, Luus F, Khan N, Makondo N, Akhalwaya IY, Qian H, Fagin R, Barahona F, Sharma U, Ikbal S, Karanam H, Neelam S, Likhyan A and Srivastava S (2020) Logical Neural Networks. DOI:10.48550/arXiv.2006.13155. URL <https://arxiv.org/abs/2006.13155>. **Note:** ArXiv preprint. The arXiv record states submission to NeurIPS 2020, but acceptance or subsequent peer-reviewed publication of this specific paper could not be verified as of 12 October 2026.
- Ritz BM and Hofacker GL (1996) Informational properties of neural nets performing algorithmic and logical tasks. *Biological Cybernetics* 74(6): 549–555. DOI: 10.1007/BF00209426. URL <https://doi.org/10.1007/BF00209426>.
- Rocktäschel T and Riedel S (2016) Learning knowledge base inference with neural theorem provers. In: *Proceedings of the 5th workshop on automated knowledge base construction*. pp. 45–50. URL <https://aclanthology.org/W16-1309.pdf>.
- Rocktäschel T and Riedel S (2017) End-to-end differentiable proving. In: *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., pp. 3788–3800. URL <https://proceedings.neurips.cc/paper/2017/file/>

[b2ab001909a8a6f04b51920306046ce5-Paper.pdf](#).

- Rollini SF, Bruttomesso R, Sharygina N and Tsitovich A (2014) Resolution proof transformation for compression and interpolation. *Formal Methods in System Design* 45(1): 1–41. URL <https://arxiv.org/pdf/1307.2028>. Peer-reviewed journal article; a manuscript version is also available as arXiv:1307.2028.
- Roslan MAF, Zamri NE, Mansor MA and Kasihmuddin MSM (2023) Major 3 Satisfiability logic in Discrete Hopfield Neural Network integrated with multi-objective Election Algorithm. *AIMS Mathematics* 8(9): 22447–22482. DOI: 10.3934/math.20231145. URL <https://doi.org/10.3934/math.20231145>.
- Roslan N, Sathasivam S and Azizan FL (2024) Conditional random k satisfiability modeling for $k = 1, 2$ (CRAN2SAT) with non-monotonic Smish activation function in discrete Hopfield neural network. *AIMS Mathematics* 9(2): 3911–3956. DOI: 10.3934/math.2024193. URL <https://doi.org/10.3934/math.2024193>.
- Roth D (1996) A Connectionist Framework for Reasoning: Reasoning with Examples. In: *Proceedings of the Thirteenth National Conference on Artificial Intelligence*. AAAI, AAAI Press, pp. 1256–1261. URL <https://cdn.aaai.org/AAAI/1996/AAAI96-186.pdf>.
- Rusdi NA, Zamri NE, Kasihmuddin MSM, Romli NA, Manoharam G, Abdeen S and Mansor MA (2024) Synergizing intelligence and knowledge discovery: Hybrid black hole algorithm for optimizing discrete Hopfield neural network with negative based systematic satisfiability. *AIMS Mathematics* 9(11): 29820–29882. DOI: 10.3934/math.20241444. URL <https://doi.org/10.3934/math.20241444>.
- Russell SJ and Norvig P (2003) *Artificial Intelligence: A Modern Approach*. 2 edition. Prentice Hall. URL <https://aima.cs.berkeley.edu/newchap07.pdf>.
- Russell SJ and Norvig P (2010) *Artificial Intelligence: A Modern Approach*. 3 edition. Upper Saddle River, NJ: Prentice Hall.
- Sander J, Yu CEJ, Jalaian B and Bastian ND (2024) Uncertainty-quantified neurosymbolic AI for open set recognition in network intrusion detection. In: *MILCOM 2024 - 2024 IEEE Military Communications Conference (MILCOM)*. pp. 13–18. DOI:10.1109/MILCOM61039.2024.10773953.
- Sathasivam S (2008) Learning in the recurrent Hopfield network. In: *2008 Fifth International Conference on Computer Graphics, Imaging and Visualisation*. IEEE, pp. 323–328. DOI:10.1109/CGIV.2008.14. URL <https://doi.org/10.1109/CGIV.2008.14>.
- Sathasivam S (2009a) Energy relaxation for Hopfield network with the new learning rule. In: *AIP Conference Proceedings*, volume 1159. American Institute of Physics, pp. 118–122. DOI:10.1063/1.3223914. URL <https://doi.org/10.1063/1.3223914>.
- Sathasivam S (2009b) Improving logic programming in Hopfield network with sign constrained. In: *2009 International Conference on Computer Technology and Development*. IEEE, pp. 161–164. DOI:10.1109/ICCTD.2009.52. URL <https://doi.org/10.1109/ICCTD.2009.52>.

[//doi.org/10.1109/ICCTD.2009.52](https://doi.org/10.1109/ICCTD.2009.52).

- Sathasivam S, Alzaeemi SA and Velavan M (2020a) Mean-Field Theory in Hopfield Neural Network for Doing 2 Satisfiability Logic Programming. *International Journal of Modern Education and Computer Science* 12(4): 27–39. DOI:10.5815/ijmecs.2020.04.03. URL <https://doi.org/10.5815/ijmecs.2020.04.03>.
- Sathasivam S, Mansor MA, Kasihmuddin MSM and Abubakar H (2020b) Election Algorithm for Random k Satisfiability in the Hopfield Neural Network. *Processes* 8(5): 568. DOI:10.3390/pr8050568. URL <https://doi.org/10.3390/pr8050568>.
- Sathasivam S, Mansor MA, Md Ismail AI, Mohd Jamaludin SZ, Mohd Kasihmuddin MS and Mamat M (2020c) Novel Random k Satisfiability for $k \leq 2$ in Hopfield Neural Network. *Sains Malaysiana* 49(11): 2847–2857. DOI:10.17576/jsm-2020-4911-23. URL <https://doi.org/10.17576/jsm-2020-4911-23>.
- Sathasivam S and Velavan M (2014) Boltzmann Machine and Hyperbolic Activation Function in Higher Order Network. *Modern Applied Science* 8(3): 140–146. DOI: 10.5539/mas.v8n3p140. URL <https://doi.org/10.5539/mas.v8n3p140>.
- Sathasivam S and Wan Abdullah WAT (2008) Logic Learning in Hopfield Networks. *Modern Applied Science* 2(3): 57–62. DOI:10.5539/mas.v2n3p57. URL <https://doi.org/10.5539/mas.v2n3p57>.
- Scarselli F, Gori M, Tsoi AC, Hagenbuchner M and Monfardini G (2009) The graph neural network model. *IEEE Transactions on Neural Networks* 20(1): 61–80. DOI: 10.1109/TNN.2008.2005605.
- Schechter J and Enoch D (2006) Meaning and justification: The case of modus ponens. *Noûs* : 687–715 URL <https://www.jstor.org/stable/4093984>.
- Seda AK (2006) On the integration of connectionist and logic-based systems. *Electronic Notes in Theoretical Computer Science* 161: 109–130. DOI:<https://doi.org/10.1016/j.entcs.2006.04.028>. URL <https://www.sciencedirect.com/science/article/pii/S1571066106004038>. **Note:** Proceedings of the Third Irish Conference on the Mathematical Foundations of Computer Science and Information Technology (MFCSIT 2004).
- Sekiyama T and Suenaga K (2018) Automated Proof Synthesis for the Minimal Propositional Logic with Deep Neural Networks. In: Ryu S (ed.) *Programming Languages and Systems: 16th Asian Symposium, APLAS 2018, Wellington, New Zealand, December 2–6, 2018, Proceedings, Lecture Notes in Computer Science*, volume 11275. Cham: Springer International Publishing, pp. 309–328. DOI:10.1007/978-3-030-02768-1_17. URL https://doi.org/10.1007/978-3-030-02768-1_17.
- Selman B, Kautz HA, Cohen B et al. (1993) Local search strategies for satisfiability testing. *Cliques, coloring, and satisfiability* 26: 521–532. URL http://eda.ee.ucla.edu/member_only/FPGA/SAT-Mapping/walksat-dimacs93.pdf.
- Selman B, Levesque HJ and Mitchell D (1992) A new method for solving hard satisfiability problems. In: *Proceedings of AAAI*. Menlo Park, CA / Cambridge,

MA: AAAI Press, pp. 440–446. URL <https://dl.acm.org/doi/10.5555/1867135.1867203>.

Selsam D and Bjørner N (2019) Guiding high-performance SAT solvers with unsat-core predictions. In: *Theory and Applications of Satisfiability Testing – SAT 2019, Lecture Notes in Computer Science*, volume 11628. Springer, pp. 336–353. DOI:10.1007/978-3-030-24258-9_24. URL https://doi.org/10.1007/978-3-030-24258-9_24. **Note:** Peer-reviewed conference paper published in the SAT 2019 proceedings. An author version is available as arXiv:1903.04671.

Selsam D, Lamm M, Bünz B, Liang P, de Moura L and Dill DL (2019) Learning a sat solver from single-bit supervision. In: *International Conference on Learning Representations (ICLR 2019)*. URL https://openreview.net/forum?id=HJMC_ia5tm. Last Modified: 08th February 2026.

Shannon CE (1948) A mathematical theory of communication. *Bell System Technical Journal* 27(3): 379–423. URL <https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6773024>.

Shi S, Chen H, Ma W, Mao J, Zhang M and Zhang Y (2020) Neural logic reasoning. In: *Proceedings of the 29th ACM International Conference on Information & Knowledge Management, CIKM '20*. Association for Computing Machinery, pp. 1365–1374. DOI:10.1145/3340531.3411949. URL <https://doi.org/10.1145/3340531.3411949>.

Shi Z, Li M, Liu Y, Khan S, Huang J, Zhen HL, Yuan M and Xu Q (2023) SATformer: Transformer-Based UNSAT Core Learning. In: *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*. IEEE, pp. 1–4. DOI:10.1109/ICCAD57390.2023.10323731. URL <https://doi.org/10.1109/ICCAD57390.2023.10323731>.

Sianipar HF, Zaini NNY, Kasihmuddin MSM, Mansor MA and Sathasivam S (2020) Hybrid ant colony optimization for even-2 satisfiability programming in hopfield neural network. *AIP Conference Proceedings* 2266(1): 040014. DOI:10.1063/5.0018264. URL <https://doi.org/10.1063/5.0018264>.

Silva A and Gombolay M (2021) Encoding human domain knowledge to warm start reinforcement learning. In: *Proceedings of the AAAI conference on artificial intelligence*, volume 35. pp. 5042–5050. URL <https://ojs.aaai.org/index.php/AAAI/article/view/16638>.

Silver D, Schrittwieser J, Simonyan K, Antonoglou I, Huang A, Guez A, Hubert T, Baker L, Lai M, Bolton A et al. (2017) Mastering the game of go without human knowledge. *nature* 550(7676): 354–359. URL <https://www.nature.com/articles/nature24270.pdf>.

Smith P (2020) Soundness and completeness. In: *An Introduction to Formal Logic*. Cambridge University Press, pp. 402–411. URL https://www.logicmatters.net/resources/pdfs/IFL2_LM.pdf.

- Someetheram V, Marsani MF, Mohd Kasihmuddin MS, Zamri NE, Muhammad Sidik SS, Mohd Jamaludin SZ and Mansor MA (2022) Random maximum 2 satisfiability logic in discrete Hopfield neural network incorporating improved election algorithm. *Mathematics* 10(24): 4734. DOI:10.3390/math10244734. URL <https://doi.org/10.3390/math10244734>.
- Spears WM (1996) A NN algorithm for Boolean satisfiability problems. In: *Proceedings of the IEEE International Conference on Neural Networks*, volume 2. IEEE, pp. 1121–1126. DOI:10.1109/ICNN.1996.549055. URL <https://doi.org/10.1109/ICNN.1996.549055>.
- Staffa M, Rossi S, Gregorio MD and Burattini E (2011) Thresholds tuning of a neuro-symbolic net controlling a behavior-based robotic system. In: Verleysen M (ed.) *ESANN 2011 Proceedings: European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*. Bruges, Belgium: i6doc.com. ISBN 978-2-87419-044-5, pp. 159–164. URL <http://www.i6doc.com/en/livre/?GCOI=28001100817300>.
- Suda M (2021) Improving enigma-style clause selection while learning from history. In: Platzter A and Sutcliffe G (eds.) *Automated Deduction – CADE 28*. Cham: Springer International Publishing. ISBN 978-3-030-79876-5, pp. 543–561. URL https://link.springer.com/chapter/10.1007/978-3-030-79876-5_31.
- Suda M (2025) Efficient neural clause-selection reinforcement. URL <https://arxiv.org/abs/2503.07792>. Cited as an arXiv manuscript; acceptance or publication in a peer-reviewed venue has not been verified.
- Sun Y, Ye F, Chen Z, Wei K and Cai S (2026) Discovering heuristics in a complex SAT solver with large language models. *Nature Communications* 17: 9008. DOI:10.1038/s41467-026-74949-2. URL <https://www.nature.com/articles/s41467-026-74949-2>.
- Tambouratzis T (1995) Simulated annealing artificial neural networks for the satisfiability (SAT) problem. In: Pearson DW, Steele NC and Albrecht RF (eds.) *Artificial Neural Nets and Genetic Algorithms*. Vienna: Springer, pp. 340–343. DOI:10.1007/978-3-7091-7535-4_89. URL https://doi.org/10.1007/978-3-7091-7535-4_89.
- Tambouratzis T (2001) An artificial neural network satisfiability tester. *International Journal of Intelligent Systems* 16(12): 1357–1375. DOI:<https://doi.org/10.1002/int.1064>. URL <https://onlinelibrary.wiley.com/doi/abs/10.1002/int.1064>.
- Tönshoff J, Kisin B, Lindner J and Grohe M (2023) One Model, Any CSP: Graph Neural Networks as Fast Global Search Heuristics for Constraint Satisfaction. In: *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*. pp. 4280–4288. DOI:10.24963/ijcai.2023/476. URL <https://www.ijcai.org/proceedings/2023/476>. **Note:** Verified IJCAI 2023 proceedings publication; official proceedings manuscript consulted. Earlier preprint: arXiv:2208.10227 (2022); counted once under 2023.

- 4214 Towell GG and Shavlik JW (1994) Knowledge-based artificial neural net-
4215 works. *Artificial Intelligence* 70(1): 119–165. DOI:[https://doi.org/10.](https://doi.org/10.1016/0004-3702(94)90105-8)
4216 1016/0004-3702(94)90105-8. URL [https://www.sciencedirect.com/science/](https://www.sciencedirect.com/science/article/pii/0004370294901058)
4217 [article/pii/0004370294901058](https://www.sciencedirect.com/science/article/pii/0004370294901058).
- 4218 Towell GG, Shavlik JW and Noordewier MO (1990) Refinement of approximate domain
4219 theories by knowledge-based neural networks. In: *Proceedings of the eighth*
4220 *National conference on Artificial Intelligence (AAAI-90)*. pp. 861–866. URL [https:](https://dl.acm.org/doi/abs/10.5555/1865609.1865629)
4221 [//dl.acm.org/doi/abs/10.5555/1865609.1865629](https://dl.acm.org/doi/abs/10.5555/1865609.1865629).
- 4222 Townsend J, Chaton T and Monteiro JM (2019) Extracting relational explanations
4223 from deep neural networks: A survey from a neural-symbolic perspective. *IEEE*
4224 *transactions on neural networks and learning systems* 31(9): 3456–3470. URL
4225 <https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8889997>.
- 4226 Tran SN (2017) Unsupervised neural-symbolic integration URL [https://arxiv.org/](https://arxiv.org/abs/1706.01991)
4227 [abs/1706.01991](https://arxiv.org/abs/1706.01991). **Note:** ArXiv preprint. Listed in the author-uploaded record as
4228 a contribution to the IJCAI 2017 Workshop on Explainable AI. Workshop peer-
4229 review status and subsequent journal or main-conference publication could not be
4230 independently verified as of 12 October 2026.
- 4231 Tran SN (2018) Propositional knowledge representation and reasoning in restricted
4232 Boltzmann machines. URL <https://arxiv.org/abs/1705.10899v3>. **Note:** ArXiv
4233 preprint, version 3, revised 29 May 2018 (first posted 31 May 2017). No peer-
4234 reviewed publication or acceptance of this specific preprint could be verified as of
4235 12 October 2026.
- 4236 Tran SN and d’Avila Garcez A (2021) Logical boltzmann machines. URL [https:](https://arxiv.org/pdf/2112.05841)
4237 [//arxiv.org/pdf/2112.05841](https://arxiv.org/pdf/2112.05841). **Note:** arXiv preprint. The LBM method was
4238 subsequently developed in the peer-reviewed AAAI 2023 paper “Neurosymbolic
4239 Reasoning and Learning with Restricted Boltzmann Machines”, pp. 6558–6565,
4240 doi:10.1609/aaai.v37i5.25806.
- 4241 Tran SN and Garcez Ad (2023) Neurosymbolic reasoning and learning with restricted
4242 boltzmann machines 37(5): 6558–6565. URL [https://ojs.aaai.org/index.php/](https://ojs.aaai.org/index.php/AAAI/article/view/25806)
4243 [AAAI/article/view/25806](https://ojs.aaai.org/index.php/AAAI/article/view/25806).
- 4244 Trevisan L (2001) Notes on the np-completeness of circuit-SAT. URL [https://theory.](https://theory.stanford.edu/~trevisan/cs154-12/notenp.pdf)
4245 [stanford.edu/~trevisan/cs154-12/notenp.pdf](https://theory.stanford.edu/~trevisan/cs154-12/notenp.pdf).
- 4246 Tönshoff J and Grohe M (2026) Learning from Algorithm Feedback: One-Shot SAT
4247 Solver Guidance with GNNs. In: *The Fourteenth International Conference*
4248 *on Learning Representations*. ICLR. URL [https://openreview.net/forum?id=](https://openreview.net/forum?id=NfWrL0Knfk)
4249 [NfWrL0Knfk](https://openreview.net/forum?id=NfWrL0Knfk).
- 4250 Unno H, Torii S and Sakamoto H (2017) Automating induction for solving horn
4251 clauses. In: *International Conference on Computer Aided Verification (CAV),*
4252 *Lecture Notes in Computer Science*, volume 10427. Springer, pp. 571–591. DOI:
4253 10.1007/978-3-319-63390-9_30.

- Vaezipoor P, Lederman G, Wu Y, Grosse R and Bacchus F (2020) Learning clause deletion heuristics with reinforcement learning. In: *Informal Proceedings of the 5th Conference on Artificial Intelligence and Theorem Proving (AITP 2020)*. pp. 85–87. URL https://aitp-conference.org/2020/abstract/paper_25.pdf.
- Velavan M, bin Yahya ZR, bin Abdul Halif MN and Sathasivam S (2015) Boltzmann machine and hyperbolic activation function in higher order neuro symbolic integration. *International Journal of Computational and Electronics Aspects in Engineering* 1(2): 63–69. **Note:** Revised in 2020.
- Velavan M, bin Yahya ZR, bin Abdul Halif MN and Sathasivam S (2016) Mean field theory in doing logic programming using Hopfield network. *Modern Applied Science* 10(1): 154. DOI:10.5539/mas.v10n1p154. URL <https://doi.org/10.5539/mas.v10n1p154>.
- Vilela IMO and Lima PMV (2001) Conjecturing the cognitive plausibility of an ANN theorem-prover. In: Mira J and Prieto A (eds.) *Connectionist Models of Neurons, Learning Processes, and Artificial Intelligence, Lecture Notes in Computer Science*, volume 2084. Springer Berlin Heidelberg, pp. 822–829. DOI: 10.1007/3-540-45720-8_99. URL https://doi.org/10.1007/3-540-45720-8_99.
- Wan Z, Liu CK, Yang H, Li C, You H, Fu Y, Wan C, Krishna T, Lin Y and Raychowdhury A (2024) Towards cognitive AI systems: A survey and prospective on neuro-symbolic AI. *IEEE Circuits and Systems Magazine* 24(2): 28–61. URL <https://arxiv.org/abs/2401.01040>. Peer-reviewed publication; an earlier version was posted as arXiv:2401.01040.
- Wan Abdullah WAT (1992) Logic programming on a neural network. *International Journal of Intelligent Systems* 7(6): 513–519. DOI:10.1002/int.4550070604. URL <https://doi.org/10.1002/int.4550070604>.
- Wan Abdullah WAT (1993) The logic of neural networks. *Physics Letters A* 176(3–4): 202–206. DOI:10.1016/0375-9601(93)91035-4. URL [https://doi.org/10.1016/0375-9601\(93\)91035-4](https://doi.org/10.1016/0375-9601(93)91035-4).
- Wan Abdullah WATb (1996) Logic Programming in Neural Networks. *Malaysian Journal of Computer Science* 9(1): 1–5. URL <https://ejournal.um.edu.my/index.php/MJCS/article/view/2888>.
- Wang F and Rompf T (2018) From Gameplay to Symbolic Reasoning: Learning SAT Solver Heuristics in the Style of Alpha(Go) Zero. URL <https://arxiv.org/abs/1802.05340>. **Note:** Research preprint / ICLR 2018 workshop-track submission; acceptance and peer-reviewed publication have not been independently verified. Consulted arXiv:1802.05340v1, first posted 14 February 2018, and the corresponding four-page OpenReview manuscript under the shorter title From Gameplay to Symbolic Reasoning. Counted once.
- Wang PW, Donti P, Wilder B and Kolter Z (2019) Satnet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. In: *International Conference on Machine Learning*. PMLR, pp. 6545–6554. URL <https://>

proceedings.mlr.press/v97/wang19e/wang19e.pdf.

Wang W, Hu Y, Tiwari M, Khurshid S, McMillan K and Miikkulainen R (2024a) NeuroBack: Improving cdcl sat solving using graph neural networks. In: *The Twelfth International Conference on Learning Representations*. URL <https://openreview.net/forum?id=samyfu6G93>.

Wang W, Yang Y and Wu F (2024b) Towards data-and knowledge-driven ai: a survey on neuro-symbolic computing. *IEEE Transactions on Pattern Analysis and Machine Intelligence* URL <https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=10721277>.

Wang Y, Liang X, Hei X, Zhu L and Ji W (2021) A heuristic SAT problems solving method based on LSTM network. In: *2021 12th International Symposium on Parallel Architectures, Algorithms and Programming (PAAP)*. IEEE, pp. 141–145. DOI: 10.1109/PAAP54281.2021.9720482. URL <https://doi.org/10.1109/PAAP54281.2021.9720482>.

Weitze MD and Hofacker GL (1995) The implementation of propositional logic in random neural networks. In: Mira J and Sandoval F (eds.) *From Natural to Artificial Neural Computation, Lecture Notes in Computer Science*, volume 930. Berlin, Heidelberg: Springer, pp. 180–188. DOI:10.1007/3-540-59497-3_173. URL https://doi.org/10.1007/3-540-59497-3_173.

Wendelken C and Shastri L (2004) Multiple instantiation and rule mediation in shruti. *Connection Science* 16(3): 211–217. URL <https://www.tandfonline.com/doi/pdf/10.1080/09540090412331311932>.

Wetzler N, Heule MJH and Hunt WA (2014) DRAT-trim: Efficient checking and trimming using expressive clausal proofs. In: Sinz C and Egly U (eds.) *Theory and Applications of Satisfiability Testing – SAT 2014*. Springer International Publishing. ISBN 978-3-319-09284-3, pp. 422–429. URL https://link.springer.com/chapter/10.1007/978-3-319-09284-3_31.

Xie X and Sathasivam S (2026) A discrete Hopfield neural network solves 3-SAT by optimizing weights based on clause-derived energy functions. *Applied Soft Computing* 202: 115872. DOI:10.1016/j.asoc.2026.115872. URL <https://doi.org/10.1016/j.asoc.2026.115872>.

Xie X, Sathasivam S and Ma H (2024) Modeling of 3 SAT discrete Hopfield neural network optimization using genetic algorithm optimized K-modes clustering. *AIMS Mathematics* 9(10): 28100–28129. DOI:10.3934/math.20241363. URL <https://doi.org/10.3934/math.20241363>.

Xu J, Zhang Z, Friedman T, Liang Y and Van den Broeck G (2018) A semantic loss function for deep learning with symbolic knowledge. In: *International Conference on Machine Learning*. PMLR, pp. 5502–5511. URL <https://proceedings.mlr.press/v80/xu18h/xu18h.pdf>.

Yolcu E and Póczos B (2019) Learning local search heuristics for boolean satisfiability. *Advances in Neural Information Processing Systems* 32.

- URL https://proceedings.neurips.cc/paper_files/paper/2019/file/12e59a33de1bf0630f46edfe13d6ea2-Paper.pdf.
- Yu D, Yang B, Liu D, Wang H and Pan S (2023) A survey on neural-symbolic learning systems. *Neural Networks* 166: 105–126. DOI:<https://doi.org/10.1016/j.neunet.2023.06.028>. URL <https://www.sciencedirect.com/science/article/pii/S0893608023003398>.
- Yun S and Wang X (2025) Multi-step partitioning combined with SOM neural network-based clustering technique effectively improves SAT solver performance. *PeerJ Computer Science* 11. DOI:10.7717/peerj-cs.3076. URL <https://peerj.com/articles/cs-3076/>.
- Zamri NE, Azhar SA, Mansor MA, Alway A and Kasihmuddin MSM (2022a) Weighted Random k Satisfiability for $k = 1, 2$ (r2SAT) in Discrete Hopfield Neural Network. *Applied Soft Computing* 126: 109312. DOI:10.1016/j.asoc.2022.109312. URL <https://doi.org/10.1016/j.asoc.2022.109312>.
- Zamri NE, Azhar SA, Sidik SSM, Mansor MA, Kasihmuddin MSM, Pakruddin SPA, Pauzi NA and Nawi SNM (2022b) Multi-discrete genetic algorithm in Hopfield neural network with weighted random k satisfiability. *Neural Computing and Applications* 34(21): 19283–19311. DOI:10.1007/s00521-022-07541-6. URL <https://doi.org/10.1007/s00521-022-07541-6>.
- Zamri NE, Mansor MA, Kasihmuddin MSM, Alway A and Karim SA (2022c) A comparative analysis of satisfiability logical structure in Discrete Hopfield Neural Network. In: *AIP Conference Proceedings*, volume 2465. AIP Publishing, p. 020008. DOI:10.1063/5.0078318. URL <https://doi.org/10.1063/5.0078318>.
- Zhai S and Ge N (2025) Learning Splitting Heuristics in Divide-and-Conquer SAT Solvers with Reinforcement Learning. In: *International Conference on Learning Representations*. ICLR. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/f5c683b93319b82689af3afc71257df2-Abstract-Conference.html.
- Zhang C, Zhang Y, Mao J, Chen W, Yue L, Bai G and Xu M (2022a) Towards better generalization for neural network-based SAT solvers. In: *Advances in Knowledge Discovery and Data Mining, Lecture Notes in Computer Science*, volume 13281. Cham: Springer International Publishing, pp. 199–210. DOI:10.1007/978-3-031-05936-0_16. URL https://doi.org/10.1007/978-3-031-05936-0_16.
- Zhang H, Li LH, Meng T, Chang KW and Broeck GVd (2022b) On the paradox of learning to reason from data. URL <https://arxiv.org/pdf/2205.11502>. **Note:** Peer-reviewed main-track paper published at IJCAI 2023. An earlier preprint appeared as arXiv:2205.11502 in 2022.
- Zhang H, Li LH, Meng T, Chang KW and Van den Broeck G (2023) On the paradox of learning to reason from data. In: Elkind E (ed.) *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence (IJCAI-23)*. International

- Joint Conferences on Artificial Intelligence Organization, pp. 3365–3373. DOI: 10.24963/ijcai.2023/375. URL <https://doi.org/10.24963/ijcai.2023/375>.
- Zhang H and Zhang J (2022) *Logic in Computer Science*. PhD Thesis. URL <https://homepage.divms.uiowa.edu/~hzhang/c188/notes/book.pdf>. **Note:** Unpublished textbook draft based on University of Iowa lecture notes.
- Zhang K and Nagamatu M (2005a) Solving satisfiability problem by parallel execution of neural networks with biases. In: Duch W, Kacprzyk J, Oja E and Zadrozny S (eds.) *Artificial Neural Networks: Formal Models and Their Applications – ICANN 2005, Part II, Lecture Notes in Computer Science*, volume 3697. Springer Berlin Heidelberg, pp. 969–974. DOI:10.1007/11550907_153. URL https://doi.org/10.1007/11550907_153.
- Zhang K and Nagamatu M (2005b) Using attenuation coefficient generating function in parallel execution of neural networks for solving SAT. *Journal of Advanced Computational Intelligence and Intelligent Informatics* 9(2): 121–126. DOI:10.20965/jaciii.2005.p0121. URL <https://doi.org/10.20965/jaciii.2005.p0121>.
- Zhang W, Sun Z, Zhu Q, Li G, Cai S, Xiong Y and Zhang L (2020) NLocalSAT: Boosting Local Search with Solution Prediction. In: *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, pp. 1177–1183. DOI:10.24963/ijcai.2020/164. URL <https://doi.org/10.24963/ijcai.2020/164>.
- Zhang Y, Zhen HL, Yuan M and Yu B (2024a) DiffSAT: Differential maxSAT layer for SAT solving. In: *Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design*. pp. 1–7. URL <https://dl.acm.org/doi/pdf/10.1145/3676536.3676748>.
- Zhang Z, Chételat D, Cotnareanu J, Ghose A, Xiao W, Zhen HL, Zhang Y, Hao J, Coates M and Yuan M (2024b) GraSS: Combining Graph Neural Networks with Expert Knowledge for SAT Solver Selection. In: *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. ACM, pp. 6301–6311. DOI: 10.1145/3637528.3671627. URL <https://doi.org/10.1145/3637528.3671627>.
- Zhang Z, Liu H, Yu Y, Chen J and Xu X (2026) Boolean satisfiability via imitation learning. In: *The Fourteenth International Conference on Learning Representations*. URL <https://openreview.net/forum?id=LNqWbY5iIf>.
- Zheng X, Li Y, Fan C, Wu H, Song X and Yan J (2024) Learning Plaintext-Ciphertext Cryptographic Problems via ANF-based SAT Instance Representation. In: *Advances in Neural Information Processing Systems*, volume 37. Curran Associates, Inc., pp. 41649–41669. DOI:10.52202/079017-1317. URL <https://doi.org/10.52202/079017-1317>.