AS^2 - Attention-Based Soft Answer Sets: An End-to-End Differentiable Neuro-*Soft*-Symbolic Reasoning Architecture

Wael Abd-Almageed
Holcombe Department of Electrical and Computer Engineering
Clemson University
wabdalm@clemson.edu

Abstract

Neuro-symbolic systems typically couple a neural perception module with a discrete symbolic solver, where exact constraint inference is intractable at training time and a discrete solver must be invoked at inference time, precluding end-to-end optimization of the model. We introduce AS^2 (Attention-Based Soft Answer Sets), a fully differentiable neuro-symbolic architecture that replaces the discrete solver with a continuous approximation of the Answer Set Programming (ASP) immediate-consequence operator T_P. AS^2 maintains per-position probability distributions over a finite symbol domain throughout the forward pass and trains end-to-end by minimizing the fixed-point residual of a probabilistic lift of T_P, thereby *differentiating through the constraint check* without invoking an external solver during either training or inference. On spatial constraint-satisfaction tasks, AS^2 replaces conventional positional embeddings entirely, and problem structure is encoded through constraint-group membership embeddings derived directly from the declarative ASP specification, making the model agnostic to arbitrary position indexing. On Visual Sudoku, AS^2 achieves 99.89% raw cell accuracy and, after greedy decoding, 100% constraint satisfaction, without invoking any external solver. On CLEVR-Hans, AS^2 achieves 99.94% on CLEVR-Hans3 and 87.77% on CLEVR-Hans7. These results demonstrate that a soft differentiable fixpoint operator, combined with constraint-aware attention and declarative constraint specification, can match or exceed pipeline and solver-based neuro-symbolic systems while maintaining full end-to-end differentiability.

1 Introduction

The integration of neural perception and symbolic reasoning has received considerable attention in recent years [Garcez et al., 2019, Manhaeve et al., 2018, Li et al., 2023]. However, neuro-symbolic pipelines either rely on intractable exact inference at training time or invoke a discrete symbolic solver at inference time, with the boundary between perception and reasoning components imposing a structural cost on both regimes. Perceptual constraint-satisfaction problems (P-CSPs) [Wang et al., 2019, Yang et al., 2020, Mulamba et al., 2024] require mapping raw sensor data (*e.g.*, images) to discrete symbolic assignments that collectively satisfy a set of declarative constraints (*e.g.*, physical laws, etc.). Neural models excel at extracting high-level representations from raw, high-dimensional inputs, while symbolic reasoners excel at enforcing global consistency through declarative constraints. The challenge of combining these two capabilities arises from the incompatibility of their computational registers, since neural inference operates over continuous distributions while symbolic reasoning requires discrete variable assignments that satisfy hard logical constraints.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

The dominant paradigm in neuro-symbolic artificial intelligence (AI) addresses P-CSPs through a *pipeline* architecture that decouples perception from reasoning, in which a neural network first processes inputs, then a discrete symbolic solver is invoked to find a globally consistent assignment. Answer Set Programming (ASP) [Brewka et al., 2011, Gelfond and Lifschitz, 1988] is the predominant symbolic formalism employed in such pipelines. ASP is a mature declarative logic-programming paradigm whose stable-model semantics support default negation, choice rules, and optimization statements, enabling compact specification of complex combinatorial constraints. Clingo [Gebser et al., 2014] is the most widely used ASP solver, while Constraint Programming SAT (CP-SAT) solvers [Mulamba et al., 2024] provide a propositional alternative when the constraint vocabulary does not require full ASP expressiveness.

Pipeline architectures introduce two structural costs that motivate this work. First, although several pipeline systems are end-to-end differentiable at training time through probabilistic stable models [Yang et al., 2020], weighted model counting [Manhaeve et al., 2018], or differentiable provenance semirings [Li et al., 2023], the underlying exact inference is in general computationally intractable (#P-hard for weighted model counting), forcing approximations that compromise the quality of the constraint signal at scale. Second, all of these systems still invoke a discrete solver at inference time, which precludes deployment in latency-sensitive or resource-constrained settings and prevents end-to-end optimization of the inference path itself. PBCS [Mulamba et al., 2024] additionally relies on a post-hoc no-good surrogate rather than a direct gradient through the constraint check. A confident perception error (*e.g.*, misclassification) can therefore render the symbolic grounding unsatisfiable, and the rich uncertainty encoded in the learned representations of the perception module is not exposed to the reasoning component during inference.

Alternative approaches attempt end-to-end differentiability by replacing the discrete solver with differentiable relaxation. SATNET [Wang et al., 2019] embeds a semi-definite programming (SDP)-relaxed Maximum Satisfiability (MaxSAT) layer as a neural module, and RRN [Palm et al., 2018] learns iterative message passing without any explicit symbolic component. However, these systems either lack formal constraint guarantees or rely on relaxations that are not sound with respect to the constraint semantics, and they do not generalize to the expressive ASP constraint vocabulary.

We introduce AS^2 (Attention-Based Soft Answer Sets), a fully differentiable neuro-*soft*-symbolic architecture built on three coupled design principles. (i) Rather than committing to discrete symbol assignments before reasoning, the model maintains continuous probability distributions over a finite symbolic vocabulary throughout the entire forward pass and enforces constraints via a differentiable loss derived from the fixed-point residual of a probabilistic lift of the ASP immediate-consequence operator T_P [van Emden and Kowalski, 1976, Takemura and Inoue, 2024], which replaces set-theoretic intersection with element-wise products and measures violation as the squared distance between the current distribution and its image under T_P, yielding a loss that is zero iff the prediction is a one-hot assignment satisfying all constraints and that admits no degenerate minimum at the uniform distribution, and gradients flow from constraint satisfaction directly into the perception encoder, and no external solver is invoked at training or inference time. (ii) On spatial constraint-satisfaction tasks, AS^2 dispenses with conventional positional embeddings entirely, where in contrast to neuro-symbolic Transformers that rely on learned or sinusoidal position encodings [McLeish et al., 2024], problem structure is encoded through *constraint-group membership embeddings* read off the declarative ASP specification, making token representations invariant to permutations of the position index and grounded in the logical structure of the problem. (iii) The probabilistic T_P surrogate is a structural translation of the symbolic operator into continuous arithmetic rather than an additional free-parameter layer, so its topology is fixed by the logic program while operator parameters are either fixed (explicit variant) or initialized from the rules and refined end-to-end (learned variant), so full differentiability does not collapse AS^2 into an unconstrained learned model, and the reasoning pathway is dictated by constraint semantics rather than learned from examples. This paper makes the following contributions.

- A fully differentiable neuro-*soft*-symbolic architecture that replaces discrete ASP solvers with a soft probabilistic T_P operator, enabling end-to-end training with constraint feedback.
- Constraint-group membership embeddings as a positional-embedding-free mechanism for encoding problem structure in attention-based models.

2 Related Work

We organize related work into three major constraint-solving families of methods that involve neural perception. The first family, pipeline neuro-symbolic systems, couples a neural perception module to a discrete symbolic solver and typically invokes the solver at inference time. The second family, end-to-end differentiable neuro-symbolic systems, replaces the discrete solver with a differentiable relaxation. The third family, perception-based constraint solving methods, supervises a neural perception module jointly with a classical constraint solver using calibration feedback computed during training, and invokes the solver again at inference.

Pipeline Neuro-Symbolic Systems. By far, the dominant paradigm in neuro-symbolic artificial intelligence (AI) couples a neural perception module with a classical symbolic solver. Several systems in this family are end-to-end differentiable at training time but still rely on a discrete solver or an intractable exact-inference procedure to compute the constraint-satisfaction signal. NeurASP [Yang et al., 2020] wraps neural network outputs as probabilistic facts in an Answer Set Programming (ASP) program and computes gradients through probabilistic stable models. Training-time gradients therefore reach the perception encoder, but inference still invokes Clingo, and the exact stable-model count grows combinatorially with the program size. DeepProbLog [Manhaeve et al., 2018, 2021a] attaches neural predicates to a ProbLog engine and computes exact gradients through weighted model counting. Weighted model counting is #P-hard in general, so the framework does not scale beyond small programs without approximation, and stable-model semantics are not supported. Scallop [Li et al., 2023] introduces differentiable Datalog with provenance semirings. Its constraint vocabulary is limited to stratified Datalog, which supports stratified default negation but excludes unstratified default negation, choice rules, and optimization statements. At scale, the provenance circuits must be approximated by top-k proofs, sacrificing soundness and completeness of the gradient signal. LTN [Badreddine et al., 2022] grounds first-order logic formulae as differentiable real-valued functions, yet the fuzzy-logic relaxation discards crisp stable-model semantics, making formal verification intractable. The shared limitation of pipeline systems is therefore not a lack of training-time differentiability, but rather (i) reliance on a discrete solver at inference time, (ii) intractable or approximated exact inference for the training signal, and (iii) restricted constraint vocabularies that exclude full ASP.

End-to-End Differentiable Neuro-Symbolic Systems. SATNet [Wang et al., 2019] embeds a differentiable semi-definite programming (SDP)-relaxed MaxSAT layer as a neural module, enabling end-to-end training. However, the SDP relaxation may certify infeasible instances as satisfiable, and its $O(n^3)$ complexity prevents scaling to large constraint systems. RRN [Palm et al., 2018] learns iterative message passing on a fully-connected cell graph, solving Sudoku without any explicit symbolic component. It serves as a strong neural baseline, yet offers no formal constraint guarantees and does not generalize to new constraint families. A-NeSI [van Krieken et al., 2023] addresses the intractability of exact inference in DeepProbLog by introducing approximate neuro-symbolic inference, at some cost to the tightness of gradient estimates. SLASH [Skryagin et al., 2022] combines probabilistic circuits with neural ASP for tractable inference via sum-product networks, yet the coupling still relies on a classical solver at test time.

The work most closely related to ours is Takemura and Inoue [2024], who propose differentiable constraint learning via the immediate-consequence operator for neural ASP. Their formulation provides the probabilistic lift of T_P that we adopt as the foundation for our constraint loss. However, Takemura and Inoue [2024] do not address the architectural question of how to propagate constraint information across positions without positional embeddings, and do not incorporate inference-time constraint propagation. AS^2 builds on their T_P formulation and extends it with constraint-group membership embeddings, a multi-layer Transformer reasoning module, and greedy constrained decoding, yielding a complete end-to-end system for perceptual constraint-satisfaction problems.

Perception-Based Constraint Solving. Mulamba et al. [2024] introduce a predict-then-optimize framework (PBCS) that trains a Cell-CNN digit classifier with calibration feedback derived from a CP-SAT solver, which is also invoked at inference time. On MNIST-based Visual Sudoku, PBCS achieves 99.4% grid accuracy, by exploiting the fact that the distribution over digit labels, when properly calibrated, provides enough soft evidence for CP-SAT to back-propagate constraint violations into the perception loss via a no-good-aware surrogate. However, PBCS has three structural limitations of the pipeline paradigm. First, the no-good surrogate is a post-hoc approximation of the true constraint

signal rather than a direct gradient through the constraint check. Second, the constraint vocabulary is limited to AllDifferent. Extending to recursive rules, default negation, or optimization statements would require a new solver entirely. Third, PBCS applies only when a complete, sound CP model exists for the target task, and a CP-SAT solver is invoked at inference time. AS^2 addresses constraint feedback through differentiable layers. Gradients flow end-to-end through the differentiable T_P residual loss, and inference is solver-free. Extending the formal coverage to default negation, choice rules, and optimization statements is left to future work.

Positional Encodings in Neuro-Symbolic Transformers. McLeish et al. [2024] encode Row-Column-Box membership as fixed binary features prepended to each token, and constraint-graph relative positional encodings [Palm et al., 2018] bias self-attention by the graph distance between constrained pairs. Yang et al. [2023] incorporate constraint-graph structure into a Recurrent Transformer for CSPs, demonstrating competitive performance on Visual Sudoku using learned positional embeddings, and Xu et al. [2025] extend this line of work with a self-supervised refinement objective that generalizes to out-of-distribution problem sizes. Kang et al. [2025] introduce a pixel-sequence benchmark explicitly designed to evaluate structural reasoning that cannot be resolved by sequential positional cues alone. Unlike these approaches, AS^2 derives its token representations directly from the declarative ASP specification, so the embedding topology is a function of the logic program rather than a hand-crafted geometric prior, and the same compilation procedure applies unchanged to any P-CSP that can be expressed in ASP. When the constraint graph is dynamic or instance-dependent (*e.g.*, in scheduling or planning domains where which variables share a constraint varies per instance), the constraint-group membership embeddings can be recompiled from the instance-specific ground program, since the compilation maps each ground atom to the set of rules in which it participates.

AS^2 takes a fundamentally different approach, replacing conventional positional embeddings entirely with *constraint-group membership embeddings*, where each token's representation is enriched by the sum of learned embeddings indexed by the constraint groups to which the token belongs (*e.g.*, row, column, and box for Sudoku). This design directly encodes the logical structure of the problem into the token representations and makes the model invariant to any re-indexing of positions that preserves constraint-group membership. To our knowledge, AS^2 is the first neuro-symbolic Transformer that achieves competitive results on Visual Sudoku without any form of conventional positional encoding.

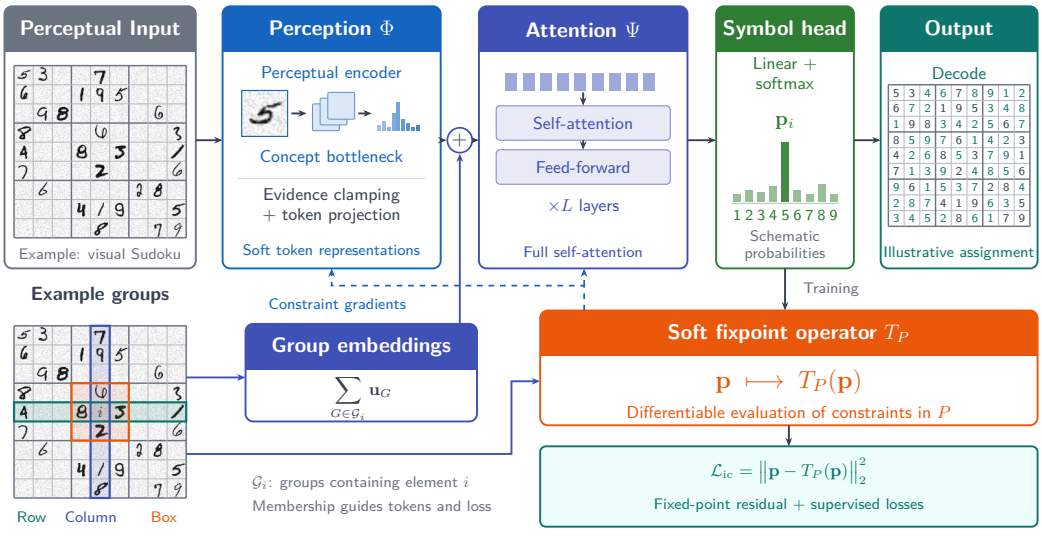


Figure 1: **AS^2** processes inputs through a neural perception module and transformer encoder, producing distributions that feed a differentiable soft fixpoint operator that reasons over symbolic concept rules.

3 Attention-Based Soft Answer Sets (AS^2)

3.1 Problem Formulation and Design Principles

In *perceptual constraint-satisfaction* problems (P-CSPs), a model observes n raw perceptual signals $\mathbf{X} = (X_1, \ldots, X_n)$ and must assign each sample i a symbol from a finite domain $\Sigma = \{s_1, \ldots, s_k\}$, subject to a set of declarative integrity constraints $\mathcal{C}$ specified as a logic program P over ground atoms drawn from a Herbrand structure over Σ. A binary evidence mask $\mathbf{m} \in \{0, 1\}^n$ identifies samples whose assignments are directly observable ($m_i = 1$, *evidence*) versus samples that must be inferred ($m_i = 0$, *latent*). The ground-truth assignment is $\mathbf{y} \in \Sigma^n$ and the task is to predict $\hat{\mathbf{y}} \in \Sigma^n$ that (i) agrees with all evidence samples and (ii) is an answer set of P, satisfying every constraint in $\mathcal{C}$. This formulation is generic, since any task in which perceptual evidence must be jointly mapped to a finite symbolic configuration that obeys declarative constraints fits within it.

As illustrated in Fig. 1, AS^2 is a neuro-symbolic constraint-satisfaction architecture in which perception, attention, and soft-symbolic reasoning layers are coupled through a single differentiable computation graph. AS^2 realizes three fundamental properties. First, gradients must flow from symbolic reasoning to perception, so perception is shaped by reasoning rather than learned in isolation. Second, integrity constraints are declaratively specified. Third, logic operators are represented as differentiable layers, and therefore no external solver is needed at either training or inference time.

3.2 AS^2 Architecture

Declarative Constraint Specification. Constraints are *declaratively* specified in standard ASP syntax [Gebser et al., 2014], using notation compatible with symbolic solvers (*e.g.*, Clingo.) The user specifies what constitutes a valid assignment, and the framework compiles each integrity constraint $:- \ \phi$ into the soft computation graph. Compilation proceeds by extracting from $\mathcal{C}$ a collection of *constraint groups* $\mathcal{G} = \{G_1, \ldots, G_M\}$, where each $G \subseteq \{1, \ldots, n\}$ is a subset of samples whose assignments must be jointly consistent with respect to a particular symbol-level condition. The collection $\mathcal{G}$ is the bridge between the symbolic specification and the neural representation.

Perception. Perception maps raw perceptual signal X_i to a continuous embedding $\mathbf{h}_i \in \mathbb{R}^{d_{\text{model}}}$ and then to pre-reasoning logits $\mathbf{z}_i^{\text{pre}} \in \mathbb{R}^k$ through a *concept bottleneck head* [Koh et al., 2020], which is a single linear layer that provides an interpretable interface between perception and reasoning.

Evidence Clamping. Evidence clamping replaces the soft distribution of observed samples ($m_i = 1$) with the one-hot ground-truth label during training, or with the perception module's argmax prediction during inference, anchoring the latent representations at known samples. The generated k-dimensional soft distributions are then projected back to d_{model} to form the input tokens of the reasoning module.

Constraint-Group Membership. Instead of using conventional positional embeddings to encode spatial structure, AS^2 encodes problem structure through *constraint-group membership embeddings*. Each token's representation is enriched by the element-wise sum of learned embedding vectors indexed by the constraint groups in $\mathcal{G}$ to which the token belongs. This design encodes the logical structure of the constraint program into the token representations, so that two samples share a high degree of representational similarity if and only if they participate in the same constraint groups. The model is therefore invariant to any re-indexing of samples that preserves constraint-group membership, and the embeddings do not impose any arbitrary sequential ordering on the problem variables.

Reasoning. The n tokens, enriched with constraint-group membership embeddings, are processed by a multi-layer Transformer encoder [Vaswani et al., 2017] with pre-norm (LayerNorm before attention and feedforward), Gaussian Error Linear Unit (GELU) activation, and full self-attention. Self-attention is the natural operator at this stage because every sample must reconcile its own perceptual evidence with the assignments at every other sample that shares a constraint group, and a single attention layer aggregates evidence across all samples simultaneously. Stacking attention layers therefore allows the latent representations to collectively converge toward a globally consistent assignment without ever committing to a discrete symbol. The output head projects each token to post-reasoning *logits* $\mathbf{z}_i^{\text{post}} \in \mathbb{R}^k$, which are passed through *softmax* to produce the per-sample distributions $\mathbf{p} \in [0, 1]^{n \times k}$.

Soft-Logic Layer. At this point we have soft per-sample distributions that are coherent with the perceptual evidence but not yet guaranteed to satisfy the integrity constraints in $\mathcal{C}$. We need a mechanism that pushes $\mathbf{p}$ toward configurations that are simultaneously valid one-hot assignments *and* consistent with $\mathcal{C}$, while remaining differentiable so that gradients can flow back through the reasoning and perception modules. The soft fixpoint layer realizes this mechanism by lifting the *immediate-consequence operator* T_P of the constraint program to probability distributions. For a definite logic program P, the least fixed point of the classical T_P is its unique answer set. A probability tensor $\mathbf{p}$ satisfies all constraints if and only if it is a fixed point of the soft lift of T_P. Training therefore minimizes the residual $\|\mathbf{p} - T_P(\mathbf{p})\|$ together with a supervised perception loss, and constraint satisfaction emerges as a by-product of driving $\mathbf{p}$ toward fixed points of T_P. The same lifting recipe extends to qualitatively different rule families, as demonstrated in Section 4.2 for existential conjunctive rules.

Probabilistic Lift of the Immediate-Consequence Operator. Let P be a definite logic program and T_P its *immediate-consequence operator* [van Emden and Kowalski, 1976], the function that maps a Herbrand interpretation I to the set of atoms whose bodies are satisfied under I. The least fixed point $\mathrm{lfp}(T_P)$ equals the unique minimal Herbrand model of P, and a ground atom a is a logical consequence of P if and only if $a \in \mathrm{lfp}(T_P)$. The integrity constraints in $\mathcal{C}$ take the form $:- \ \phi$, which are violated whenever ϕ holds under the predicted assignment, and each such constraint induces the constraint group $G \subseteq \{1, \ldots, n\}$. Given per-element distributions $\mathbf{p} \in [0,1]^{n \times k}$ with $\sum_s p_{i,s} = 1$ for every cell i, the probabilistic T_P operator applied to a mutual-exclusivity group $G \in \mathcal{G}$ is defined element- and symbol-wise as shown in Eq. (1),

$$T_P(\mathbf{p})_{i,s} \ = \ p_{i,s} \cdot \prod_{j \in G,\, j \neq i} \big(1 - p_{j,s}\big), \qquad i \in G, \ s \in \Sigma. \tag{1}$$

Intuitively, element i can *claim* symbol s only if it already assigns probability mass to s *and* no competing element of G does so. For normalized inputs and $|G| = k$, the fixed-point condition $\mathbf{p} = T_P(\mathbf{p})$ holds if and only if $\mathbf{p}$ is a valid one-hot assignment in which every group constraint is satisfied. The softmax outputs enforce this normalization, excluding the all-zero tensor from the admissible domain.

3.3 The T_P Fixed-Point Loss

Naive constraint penalties for exclusivity, such as the squared-sum loss $(\sum_i p_{i,s} - 1)^2$ [Wang et al., 2019], have a degenerate global minimum at the uniform distribution $p_{i,s} = 1/k$, which satisfies no actual constraint. The AS^2 constraint loss avoids this pathology by directly minimizing the fixed-point residual of T_P across all constraint groups, as defined in Eq. (2),

$$\mathcal{L}_{\text{ic}}(\mathbf{p}) \ = \ \sum_{G \in \mathcal{G}} \sum_{i \in G} \sum_{s \in \Sigma} \big(p_{i,s} - T_P(\mathbf{p})_{i,s}\big)^2. \tag{2}$$

At the uniform distribution, $T_P(\mathbf{p})_{i,s} = (1/k) \cdot ((k-1)/k)^{|G|-1} \neq 1/k$ for any $k \geq 2$ and $|G| \geq 2$, so $\mathcal{L}_{\text{ic}} > 0$, where the group size $|G|$ and domain size k need not coincide in general. Therefore, the degenerate minimizer of the squared-sum loss is *not* a minimizer of $\mathcal{L}_{\text{ic}}$. Conversely, at any valid one-hot assignment the product in Equation (1) collapses to $p_{i,s} \cdot 1^{|G|-1} = p_{i,s}$, giving $\mathcal{L}_{\text{ic}} = 0$. The entire computation is differentiable and fully vectorized in log-space for numerical stability [Takemura and Inoue, 2024], so no solver is invoked during either training or inference. An alternative arithmetic instantiation of T_P for global-sum constraints is described in Appendix A.6.

4 Experimental Evaluation

We apply AS^2 to three neuro-symbolic benchmarks spanning different constraint types and input modalities. In this section, we discuss the results of applying AS^2 to Visual Sudoku [Wang et al., 2019] and CLEVR-Hans [Stammer et al., 2021]. Visual Sudoku is governed by mutual-exclusivity rules over a fixed grid of grayscale digits. For Sudoku, T_P is fixed throughout training. CLEVR-Hans is governed by existential conjunctive rules over a variable-size object set in color 3-D scenes. Its Explicit variant fixes the ground-truth rules, while its Learned-operator variant, used only for CLEVR-Hans, updates rule-initialized weights end-to-end. Across these benchmarks, AS^2 uses clue and blank-cell digit

labels for Sudoku, scene-class and per-object attribute labels for CLEVR-Hans, and digit and sum labels for MNIST Addition. Table 4 in the supplementary material details this supplied information, which qualifies comparisons with published methods. Full dataset statistics and AS^2 instantiations as well as results on MNIST Addition [Li et al., 2023] are deferred to Appendices A.3–A.6.

4.1 Visual Sudoku

A 9×9 Sudoku puzzle is rendered as a 252×252 grayscale image by placing MNIST images in *clue* cells and leaving blank non-clue cells [Wang et al., 2019, Yang et al., 2020]. A model receives the full board image and must predict all 81 cell values while satisfying three families of Sudoku constraints, requiring that each row, each column, and each 3×3 box contains the digits 1–9 exactly once. We use 9,000 / 1,000 / 1,000 boards for train / validation / test with approximately 45 clue cells per board. This benchmark requires genuine constraint reasoning over 27 interacting constraint groups and cannot be solved by perception alone. The instantiation of AS^2 for Visual Sudoku is described in detail in Appendix A.4, as well as the complete declarative constraint specification (Fig. 2).

Metrics. For Visual Sudoku, we report symbolic accuracy (Sym-Acc), defined as the fraction of individual cells predicted correctly, board accuracy (Board-Acc), defined as the fraction of boards in which *all* 81 cells are correct, and constraint satisfaction rate (CSR), defined as the fraction of boards satisfying all row, column, and box constraints as measured algebraically from the model's output distributions.

Baselines. We compare against neuro-symbolic systems that reported results on Visual Sudoku. SATNet [Wang et al., 2019] introduced the benchmark and embeds a differentiable SDP-relaxed MaxSAT layer. PBCS [Mulamba et al., 2024] applies perception-based CP-SAT constraint solving. We also include Pipeline [Yang et al., 2020], our reproduction of NeurASP-style perception followed by Clingo post-processing as a reference for solver-augmented inference. RRN [Palm et al., 2018] is evaluated on symbolic (digit-representation) Sudoku only and does not report Visual Sudoku results.

Results. As shown in Table 1, AS^2 achieves a cell accuracy (Sym-Acc) of 99.89%, reflecting high accuracy on clue cells, where the prediction is anchored to the convolutional neural network (CNN) digit classifier's argmax, and near-perfect accuracy on blank cells where the Transformer must infer the correct digit from constraint propagation alone. The unaided board accuracy (raw argmax with no post-processing) is 95.60%, reflecting residual constraint-satisfaction failures concentrated in blank cells where the Transformer has not fully converged to a valid fixed point. Iterative T_P refinement with $K=10$ steps (Eq. 4) raises the constraint satisfaction rate to 98.7%, demonstrating that the trained distributions already encode sufficient constraint information to recover approximately 70.45% of the argmax failures through soft propagation alone. Greedy constrained decoding (Appendix A.4) raises board accuracy to 100% for the model-plus-decoder combination, improving on raw argmax by 4.4 percentage points, while achieving a CSR of 100.0% across all 1,000 test boards. Neither procedure calls an external solver. The greedy decoder nevertheless makes discrete constraint-based assignments, and it does not include any form of backtracking or search, compared to traditional solver-augmented pipelines. PBCS [Mulamba et al., 2024] reports the highest board accuracy of 99.4% by coupling a perception module with a full CP-SAT solver at inference time, although it does not report constraint satisfaction metrics and requires a complete constraint model at test time. RRN [Palm et al., 2018] is evaluated on symbolic Sudoku only and is therefore not directly comparable on the perceptual task. AS^2 achieves 100% board accuracy and 100% constraint satisfaction without any external solver, surpassing all previously reported results.

4.2 CLEVR-Hans

CLEVR-Hans [Stammer et al., 2021] is a visual concept-classification benchmark in which a model receives an RGB image of a synthetic 3-D scene containing between 3 and 10 objects and must assign the scene to one of K mutually exclusive concept classes. CLEVR-Hans3 ($K=3$) and CLEVR-Hans7 ($K=7$) share the same first three classes. CLEVR-Hans7 adds four additional classes including a disjunctive class and a spatial-relation class. Classes 0–1 in CLEVR-Hans3 (and classes 0–1 and 4 in CLEVR-Hans7) are *confounded*, where a spurious attribute correlation is present in the training and validation splits but absent in the test split. Correct test-time classification therefore requires learning the true symbolic rule rather than the spurious shortcut. We include these benchmarks to demonstrate

Table 1: Results on Visual Sudoku. **Bold** = best, underline = second-best. §PIPELINE CSR reflects Clingo post-processing, not learned satisfaction. We report three AS^2 inference regimes on the same 1,000 test boards: raw argmax of the Transformer output (pre-decoding), iterative T_P refinement with $K{=}10$ steps applied to the soft distributions (Eq. 4), and greedy constrained decoding (Appendix A.4). None of the three regimes invokes an external solver. [a]Evaluated on SATNet-V. [b]Evaluated on RRN-V. Both Recurrent Transformer results use the ungrounded setting without clue-digit labels.

Method	**Clues**	SYM-ACC ↑	BOARD-ACC ↑	CSR ↑
SATNET [Wang et al., 2019]	N/A	N/A	63.2	N/A
PIPELINE § [Yang et al., 2020]	N/A	60.51	0.00	100.00
PBCS [Mulamba et al., 2024]	N/A	N/A	<u>99.4</u>	N/A
Recurrent Transformer[a] [Yang et al., 2023]	31–42	N/A	93.5	N/A
Recurrent Transformer[b] [Yang et al., 2023]	17–34	N/A	75.6	N/A
AS^2 (raw argmax)	≈ 45	99.89	95.60	95.60
AS^2 ($+ K{=}10$ T_P refinement)	≈ 45	99.89	98.70	98.70
AS^2 ($+$ greedy constrained decoding)	≈ 45	**100.0**	**100.0**	**100.0**

that the differentiable T_P operator generalizes to existential conjunctive rules over multi-object scenes, a constraint family qualitatively distinct from mutual-exclusivity constraints. Benchmark details, concept rules, confounder specifications, and AS^2 instantiation are given in Appendix A.5.

Metrics. For CLEVR-Hans3 and CLEVR-Hans7, we report scene accuracy (SCENE-ACC), which is the fraction of test scenes classified into the correct concept class and, because each dataset is class-balanced, equals macro-averaged per-class accuracy.

Baselines. For the CLEVR-Hans benchmarks, we compare against all four methods reported by Stammer et al. [2021]. CNN-DEFAULT is a Residual Network (ResNet)-34 trained with standard cross-entropy and no symbolic component, with Gradient-weighted Class Activation Mapping (Grad-CAM) saliency maps produced for interpretability analysis. CNN-XIL extends CNN-DEFAULT with Explanatory Interactive Learning (XIL), in which a human oracle corrects the model's Grad-CAM explanations during training, so this method is not fully automated. NESY-CL couples a CNN object extractor with a concept-level symbolic classifier trained with cross-entropy only, and is the fair fully-automated neuro-symbolic comparison. NESY-XIL † extends NESY-CL with human-in-the-loop corrections applied to symbolic concept explanations.

Results. Table 2 reports SCENE-ACC on the CLEVR-Hans3 and CLEVR-Hans7 validation and test sets for all four AS^2 variants (Appendix A.5) alongside baselines from Stammer et al. [2021]. All four AS^2 variants achieve perfect validation accuracy (100.0%) on CLEVR-Hans3, regardless of perception module or T_P operator type, confirming that the training regime is stable across all configurations. Explicit variants maintain near-perfect test accuracy ($99.94_{\pm 0.04}\%$ and $99.76_{\pm 0.12}\%$), since the hard-coded ground-truth rules are unaffected by the spurious attribute correlations present during training. CNN-DEFAULT collapses to 70.34% on the test split, confirming that pure neural classifiers without symbolic grounding are maximally susceptible to the confounder. AS^2 Crop + Explicit at $99.94_{\pm 0.04}\%$ surpasses even the human-assisted NESY-XIL † (91.31%) by $+8.6$ percentage points, and surpasses the automated NESY-CL (81.71%) by $+18.2$ points, all without any human involvement. By contrast, learned variants collapse to $78.19_{\pm 1.22}\%$ and $72.89_{\pm 0.48}\%$ on the test split, a drop of more than twenty percentage points. This bifurcation arises because the learned concept-weight matrix $\mathbf{W}_{k,j}$ can absorb the spurious correlation during training: for C_0, the model learns that gray $\Rightarrow$ large cube (the confound) rather than the true rule, and for C_1 it learns metal $\Rightarrow$ small sphere. When the confounder is absent at test time, the learned operator produces incorrect concept scores, whereas the explicit operator remains correct by design.

On CLEVR-Hans7, the sole relevant comparison for an automated system is NESY-CL, which achieves $90.97_{\pm 0.91}\%$. AS^2 Crop + Explicit achieves $87.77_{\pm 1.50}\%$, trailing NESY-CL by 3.2 percentage points while requiring zero human involvement. NESY-XIL † achieves $94.96_{\pm 0.49}\%$ by means of a human-in-the-loop oracle, constituting privileged supervision unavailable to any automated method, and comparing against it conflates model quality with annotation cost. The NESY-CL comparison is also not on equal structural footing, since NESY-CL couples its symbolic classifier with a perception

Table 2: Scene accuracy (%) on CLEVR-Hans3 and CLEVR-Hans7. Mean ± std over three seeds. **Bold** = best automated; underline = second-best.

		CLEVR-Hans3		CLEVR-Hans7	
Method	Variant	Val↑	Test↑	Val↑	Test↑
Automated baselines (no human supervision)					
CNN-DEFAULT [Stammer et al., 2021]	N/A	$99.55_{\pm 0.10}$	$70.34_{\pm 0.30}$	$96.09_{\pm 0.19}$	$84.50_{\pm 1.04}$
CNN-XIL [Stammer et al., 2021]	N/A	$99.69_{\pm 0.08}$	$70.77_{\pm 0.37}$	$96.08_{\pm 0.25}$	$89.26_{\pm 0.29}$
NESY-CL [Stammer et al., 2021]	N/A	$98.55_{\pm 0.27}$	$81.71_{\pm 3.09}$	$96.88_{\pm 0.16}$	$\mathbf{90.97_{\pm 0.91}}$
*Human-in-the-loop baseline (†requires per-epoch human oracle corrections; **not directly comparable**)*					
NESY-XIL † [Stammer et al., 2021]	N/A	$100.00_{\pm 0.00}$	$91.31_{\pm 3.13}$	$98.76_{\pm 0.17}$	$94.96_{\pm 0.49}$
AS^2 (ours)	Crop + Explicit	$\mathbf{100.00_{\pm 0.00}}$	$\mathbf{99.94_{\pm 0.04}}$	$\mathbf{98.73_{\pm 0.05}}$	$87.77_{\pm 1.50}$
	Scene + Explicit	$\mathbf{100.00_{\pm 0.00}}$	$\underline{99.76}_{\pm 0.12}$	$98.42_{\pm 0.09}$	$86.52_{\pm 0.27}$
	Crop + Learned	$\mathbf{100.00_{\pm 0.00}}$	$78.19_{\pm 1.22}$	$98.66_{\pm 0.16}$	$82.38_{\pm 2.02}$
	Scene + Learned	$\mathbf{100.00_{\pm 0.00}}$	$72.89_{\pm 0.48}$	$98.65_{\pm 0.05}$	$83.08_{\pm 3.22}$

encoder that was *pretrained on the original CLEVR dataset* (*i.e.*, the same synthetic renderer, object palette, and attribute space as CLEVR-Hans7) and then *frozen in its entirety* before any CLEVR-Hans7 training begins [Stammer et al., 2021]. Only a lightweight Set Transformer classifier is subsequently trained on CLEVR-Hans7 labels, where the encoder is permanently shielded from the confounded objective and arrives at evaluation with in-domain, uncontaminated perceptual knowledge. AS^2, by contrast, initializes its ResNet-18 backbone from *ImageNet* weights (*i.e.*, using real images whose texture, lighting conditions, and depth cues are fundamentally different from those of synthetic 3-D renders) and trains the *entire architecture end-to-end*. The backbone, multi-layer Transformer, and T_P operator are all updated jointly against the confounded CLEVR-Hans7 objective. Every confounded gradient flows back through the full network. The 3.2% gap is within one-standard-deviation uncertainty of both estimates and contrasts with Hans3, where the explicit AS^2 *leads* all automated methods by $+18.2$ percentage points. The Hans7 shortfall plausibly reflects the residual ImageNet-to-CLEVR domain shift and the increased perception difficulty of seven finer-grained concept classes, rather than any weakness in the constraint-reasoning component. The explicit operator again generalizes substantially better to the test split than the learned operator (87.77% vs. 82.38% for Crop), replicating the confounder-robustness finding from Hans3. Within each operator type, Crop perception consistently outperforms Scene Attention (87.77% vs. 86.52% for explicit; 82.38% vs. 83.08% for learned), confirming that precise per-object Region of Interest Alignment (ROI-Align) localization is more valuable than global context for fine-grained attribute matching.

4.3 Additional Comparisons

Table 3 compares Sudoku results for ConsFormer [Xu et al., 2025], SATNET [Wang et al., 2019], Recurrent Transformer [Yang et al., 2023], and AS^2. These symbolic methods receive digit boards and report solved-instance rates on in-distribution and out-of-distribution sets. AS^2, on the other hand, receives images and achieves 98.70% board accuracy after ten soft-refinement steps. Yang et al. [2023] also report that removing learned positional embeddings on textual Sudoku reduces accuracy from 100% to 0%. This distinguishes the embedding designs, but the cross-paper accuracy comparison does not isolate their effect.

Table 3: Comparison with discrete symbolic methods. Results report solved-instance percentages for in-distribution (ID) / out-of-distribution (OOD) sets. AS^2 reports board accuracy on 1,000 visual test boards with approximately 45 clues. †Results reported by Xu et al. [2025].

Method	**Input**	**Clues / setting**	**Result (%)**
ConsFormer [Xu et al., 2025]	Digits	ID / OOD; 2,000 iterations	100 / 65.88
ConsFormer [Xu et al., 2025]	Digits	ID / OOD; 10,000 iterations	100 / 77.74
SATNET † [Wang et al., 2019]	Digits	ID / OOD	98.3 / 3.2
Recurrent Transformer† [Yang et al., 2023]	Digits	ID / OOD; 2,000 iterations	97.7 / 14.0
AS^2 (K=10 refinement)	Images	Visual test; 10 iterations	98.70

Table 6 in Appendix A.6 compares MNIST Addition with Takemura and Inoue [2024], whose Table 2 reports digit accuracy under sum-only supervision. Our two-digit model uses both digit and sum labels and achieves $99.73 \pm 0.03\%$ digit accuracy and $99.45 \pm 0.08\%$ sum accuracy.

4.4 Discussion

The empirical results validate the design of AS^2 as a constraint-satisfaction framework that leverages a differentiable probabilistic lift of the immediate-consequence operator T_P to enforce hard combinatorial constraints at test time without any external reasoning engine, symbolic-perception interface, or discrete search. The operator internalizes the constraint semantics of the declarative program during end-to-end training and uses differentiable propagation followed by discrete greedy decoding, a property that prior neuro-symbolic architectures either do not possess or achieve only by freezing perception and delegating reasoning to an oracle. The generality of AS^2 is established by empirical evaluation on several qualitatively incomparable constraint families, mutual-exclusivity over a fixed positional grid and existential conjunctive quantification over a variable-cardinality object set, and across two distinct perceptual modalities, grayscale digit recognition and multi-attribute 3-D scene understanding, confirming that the soft T_P construction is not tuned to a particular program structure but constitutes a genuinely general reasoning primitive. The implication for the design of neuro-symbolic systems is that symbolic guarantees require structural commitment, not merely initialization, and the soft T_P framework provides that commitment while remaining fully differentiable.

5 Conclusion

We introduced AS^2, a fully differentiable neuro-*soft*-symbolic architecture that replaces discrete ASP solvers with a soft probabilistic lift of the immediate-consequence operator T_P. Three principles distinguish AS^2 from prior neuro-symbolic systems. First, the architecture is entirely soft, maintaining per-position probability distributions over the symbol domain throughout the forward pass and enforcing constraints via a differentiable fixed-point residual loss that enables end-to-end gradient flow from constraint satisfaction into perception. Second, on spatial constraint-satisfaction tasks the architecture is free of conventional 1D positional embeddings. Problem structure is encoded exclusively through constraint-group membership embeddings derived from the declarative ASP specification. Third, the model approximates a soft differentiable version of the classical fixpoint operator of logic programming, providing a principled surrogate for the demonstrated constraint patterns that avoids the degenerate minima of naive penalty formulations. A natural concern that may arise about any fully differentiable architecture is whether it retains genuine symbolic character once discrete inference is replaced by continuous computation. We believe that the relevant criterion is not whether gradients are used, but rather what structure governs the reasoning computation. In a generic neural network, the reasoning transformation is parameterized and learned entirely from data, with no prior commitment to its form. In AS^2, the reasoning layer is a structural translation of the ASP immediate-consequence operator T_P whose topology is fixed by the declarative logic program. In the *explicit* variant, the operator parameters are not adjusted during training. In the *learned* variant, a small weight matrix is initialized from the rules and refined end-to-end while the operator topology is kept fixed. Symbolic structure survives relaxation in AS^2, because the governing computation encodes the constraint semantics of the program rather than empirically approximating them.

Limitations and Reproducibility. AS^2 is restricted to the definite-program fragment of ASP, so default negation, choice rules, and optimization statements are not yet covered by the soft T_P operator. Greedy constrained decoding could achieve perfect constraint satisfaction, but no theoretical guarantee exists that the learned soft distributions will universally prevent dead ends on more general problems. Code will be made available at https://huggingface.co/wabdalm/AS2.

Broader Impact. AS^2 has applications in material science and drug discovery. A potential risk is that AS^2 could be applied to automate consequential decisions without adequate human oversight, in domains such as medical diagnosis.

Acknowledgments and Disclosure of Funding

No external funding was used for this work. The authors declare no competing interests.

References

S. Badreddine, A. d. Garcez, L. Serafini, and M. Spranger. Logic tensor networks. *Artificial Intelligence*, 303:103649, 2022. doi: 10.1016/j.artint.2021.103649. URL https://doi.org/10.1016/j.artint.2021.103649. 3, 17

G. Brewka, T. Eiter, and M. Truszczyński. Answer set programming at a glance. *Communications of the ACM*, 54(12):92–103, 2011. doi: 10.1145/2043174.2043195. URL https://doi.org/10.1145/2043174.2043195. 2

A. S. d. Garcez, M. Gori, L. C. Lamb, L. Serafini, M. Spranger, and S. N. Tran. Neural-symbolic computing: An effective methodology for principled integration of machine learning and reasoning. *FLAP*, 6(4):611–632, 2019. URL https://collegepublications.co.uk/ifcolog/?00033. 1

M. Gebser, R. Kaminski, B. Kaufmann, and T. Schaub. Clingo = ASP + control: Preliminary report. In *Technical Communications of the Thirtieth International Conference on Logic Programming*, 2014. URL https://arxiv.org/abs/1405.3694. 2, 5

M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *Proceedings of the Fifth International Conference and Symposium on Logic Programming*, pages 1070–1080. MIT Press, 1988. URL https://dblp.org/rec/conf/iclp/GelfondL88. 2

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. doi: 10.1109/CVPR.2016.90. URL https://doi.org/10.1109/CVPR.2016.90. 18

K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2961–2969, 2017. doi: 10.1109/ICCV.2017.322. URL https://doi.org/10.1109/ICCV.2017.322. 17, 18

L. Kang, X. Fu, M. A. Souibgui, A. Barsky, L. Gomez, J. Vazquez-Corral, A. Fornés, E. Valveny, and D. Karatzas. A benchmark for symbolic reasoning from pixel sequences: Grid-level visual completion and correction. *Mathematics*, 13(17):2851, 2025. doi: 10.3390/math13172851. URL https://doi.org/10.3390/math13172851. 4

P. W. Koh, T. Nguyen, Y. S. Tang, S. Mussmann, E. Pierson, B. Kim, and P. Liang. Concept bottleneck models. In *Proceedings of the 37th International Conference on Machine Learning (ICML 2020)*, volume 119 of *Proceedings of Machine Learning Research*, pages 5338–5348. PMLR, 2020. URL http://proceedings.mlr.press/v119/koh20a.html. 5

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791. URL https://doi.org/10.1109/5.726791. 19

Z. Li, J. Huang, and M. Naik. Scallop: A language for neurosymbolic programming. *Proc. ACM Program. Lang.*, 7(PLDI):1463–1487, 2023. doi: 10.1145/3591280. URL https://doi.org/10.1145/3591280. 1, 2, 3, 7, 20

R. Manhaeve, S. Dumancic, A. Kimmig, T. Demeester, and L. De Raedt. DeepProbLog: Neural probabilistic logic programming. In *Advances in Neural Information Processing Systems*, volume 31, pages 3749–3759, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/dc5d637ed5e62c36ecb73b654b05ba2a-Abstract.html. 1, 2, 3, 19, 20

R. Manhaeve, S. Dumančić, A. Kimmig, T. Demeester, and L. De Raedt. Neural probabilistic logic programming in DeepProbLog. *Artificial Intelligence*, 298:103504, 2021a. doi: 10.1016/j.artint.2021.103504. URL https://doi.org/10.1016/j.artint.2021.103504. 3, 20

R. Manhaeve, G. Marra, and L. De Raedt. Approximate inference for neural probabilistic logic programming. In *Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning*, pages 475–486, 2021b. doi: 10.24963/kr.2021/45. URL https://doi.org/10.24963/kr.2021/45. 20

S. McLeish, A. Bansal, A. Stein, N. Jain, J. Kirchenbauer, B. R. Bartoldson, B. Kailkhura, A. Bhatele, J. Geiping, A. Schwarzschild, and T. Goldstein. Transformers can do arithmetic with the right embeddings. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/c35986bc1ee29b31c1011481b77fe540-Abstract-Conference.html. 2, 4

M. Mulamba, J. Mandi, A. İ. Mahmutoğulları, and T. Guns. Perception-based constraint solving for Sudoku images. *Constraints*, 29(1–2):112–151, 2024. doi: 10.1007/s10601-024-09372-9. URL https://doi.org/10.1007/s10601-024-09372-9. 1, 2, 3, 7, 8

R. Palm, U. Paquet, and O. Winther. Recurrent relational networks. In *Advances in Neural Information Processing Systems*, volume 31, pages 3368–3378, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/b9f94c77652c9a76fc8a442748cd54bd-Abstract.html. 2, 3, 4, 7

A. Skryagin, W. Stammer, D. Ochs, D. S. Dhami, and K. Kersting. Neural-probabilistic answer set programming. In *Proceedings of the 19th International Conference on Principles of Knowledge Representation and Reasoning*, pages 463–473, 2022. doi: 10.24963/kr.2022/48. URL https://doi.org/10.24963/kr.2022/48. 3

W. Stammer, P. Schramowski, and K. Kersting. Right for the right concept: Revising neuro-symbolic concepts by interacting with their explanations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3619–3629, 2021. URL https://arxiv.org/abs/2011.12854. 6, 7, 8, 9, 17, 23

A. Takemura and K. Inoue. Differentiable logic programming for distant supervision. In *Proceedings of the 27th European Conference on Artificial Intelligence (ECAI)*, volume 392 of *Frontiers in Artificial Intelligence and Applications*, pages 1301–1308. IOS Press, 2024. doi: 10.3233/FAIA240628. URL https://doi.org/10.3233/FAIA240628. 2, 3, 6, 10, 19

M. H. van Emden and R. A. Kowalski. The semantics of predicate logic as a programming language. *Journal of the ACM*, 23(4):733–742, 1976. doi: 10.1145/321978.321991. URL https://doi.org/10.1145/321978.321991. 2, 6

E. van Krieken, T. Thanapalasingam, J. M. Tomczak, F. van Harmelen, and A. ten Teije. A-NeSI: A scalable approximate method for probabilistic neurosymbolic inference. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/4d9944ab3330fe6af8efb9260aa9f307-Abstract-Conference.html. 3, 20

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html. 5

P.-W. Wang, P. Donti, B. Wilder, and Z. Kolter. SATNet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. In *Proceedings of the 36th International Conference on Machine Learning (ICML 2019)*, volume 97 of *Proceedings of Machine Learning Research*, pages 6545–6554. PMLR, 2019. URL http://proceedings.mlr.press/v97/wang19e.html. 1, 2, 3, 6, 7, 8, 9

T. Winters, G. Marra, R. Manhaeve, and L. De Raedt. DeepStochLog: Neural stochastic logic programming. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 10090–10100, 2022. doi: 10.1609/aaai.v36i9.21248. URL https://doi.org/10.1609/aaai.v36i9.21248. 19

Y. W. Xu, W. Li, S. Sanner, and E. B. Khalil. Self-supervised transformers as iterative solution improvers for constraint satisfaction. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, pages 69432–69450. PMLR, 2025. URL https://proceedings.mlr.press/v267/xu25q.html. 4, 9, 14

Z. Yang, A. Ishay, and J. Lee. NeurASP: Embracing neural networks into answer set programming. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, pages 1755–1762, 2020. doi: 10.24963/ijcai.2020/243. URL https://doi.org/10.24963/ijcai.2020/243. 1, 2, 3, 7, 8, 19, 20

Z. Yang, A. Ishay, and J. Lee. Learning to solve constraint satisfaction problems with recurrent transformer. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023. URL https://arxiv.org/abs/2307.04895. 4, 8, 9, 14

A Supplementary Material

A.1 Additional Method Details

Table 4 summarizes the training labels and symbolic rules supplied to AS^2 for each benchmark. These include supervision of individual digits or object attributes in addition to task outputs, together with fixed rules or rule-initialized weights, which must be considered when comparing methods with different access to supervision and symbolic information.

The AS^2 forward pass proceeds in five stages. (1) Perception maps each raw input element to a continuous embedding and pre-reasoning logits through a concept bottleneck head. (2) Constraint-group membership embeddings, computed from the declarative ASP specification, are added to the perception embeddings to encode problem structure without conventional positional embeddings. (3) A multi-layer Transformer encoder reasons over the resulting tokens, producing per-position post-reasoning distributions. (4) The soft logic layer applies a probabilistic lift of the immediate-consequence operator T_P that is specialized to the rule family of the benchmark, namely mutual-exclusivity for Visual Sudoku, existential conjunctive quantification for CLEVR-Hans, and arithmetic global-sum for MNIST Addition. (5) An inference-time procedure recovers a discrete prediction without invoking an external solver. The next three subsections specialize this generic pipeline to each benchmark in turn, and the order of presentation is the same throughout: detailed benchmark description, perception, constraint-group structure, reasoning module, T_P instantiation, training loss, and inference procedure.

Table 4: Supervision and symbolic information supplied to AS^2. Comparisons with published methods are not uniformly matched for supervision or rule access.

Benchmark	Supervision and supplied rules
Visual Sudoku	Clue and blank-cell digit labels. The blank-cell loss weight decays to a $0.1\times$ floor. Fixed row, column, and box constraints.
CLEVR-Hans	Scene-class and per-object attribute labels. Explicit rules are fixed from ground truth. Learned weights are rule-initialized.
MNIST Addition	Individual digit and sum labels, with the known addition relation.

For the Sudoku comparisons, Yang et al. [2023] use 9,000 training and 1,000 test boards per visual setting, while our split adds 1,000 validation boards. The symbolic comparison reported by Xu et al. [2025] uses 1,000 ID and 18,000 OOD instances.

A.2 Compute Resources

All training and evaluation were performed on a single NVIDIA H200 GPU (143 GB HBM3). Training runs for Visual Sudoku and CLEVR-Hans complete in approximately 2–4 hours per configuration. MNIST Addition runs complete in under 1 hour per scale. No multi-GPU or distributed training was used.

A.3 Benchmark Statistics

Table 5 summarizes the key statistics for all three benchmarks evaluated in this paper.

A.4 Additional Sudoku Results

AS^2 Instantiation for Visual Sudoku. The Sudoku instantiation specializes the generic AS^2 framework along four dimensions, namely the perception front-end, the constraint-group membership embeddings, the training loss, and two inference-time constraint-enforcement strategies.

Perception. The perception module applies a shared-weight three-block CNN ($32 \to 64 \to 128$ channels, max-pooling, LayerNorm) independently to each of the 81 cells extracted as 28×28 grayscale images from the 252×252 board image.

Table 5: Dataset statistics for all benchmarks evaluated in this paper. *Constraint type* classifies the symbolic constraint family. *#Groups* lists the number of constraint groups per instance. No held-out validation split is used for $\text{MNIST}_{\text{Add}}$, and the best checkpoint is selected on test accuracy following the convention of prior work.

Benchmark	**Train**	**Val**	**Test**	**Input size**	**Constraint type**	**#Groups**
Visual Sudoku						
Visual Sudoku	9,000	1,000	1,000	252×252	Latin square	27
CLEVR-Hans						
CLEVR-Hans3	9,000	2,250	2,250	320×320	Existential conj.	$3\times N_{\text{obj}}$
CLEVR-Hans7	21,000	5,250	5,250	320×320	Existential conj.	$7\times N_{\text{obj}}$
MNIST Addition						
$N{=}2$	30,000	N/A	5,000	$2\times28\times28$	Arithmetic (sum)	1
$N{=}4$	30,000	N/A	5,000	$4\times28\times28$	Arithmetic (sum)	1
$N{=}8$	30,000	N/A	5,000	$8\times28\times28$	Arithmetic (sum)	1

Constraint-Group Membership Embeddings. Three embedding tables are defined, one for each of the nine rows, nine columns, and nine 3×3 boxes, yielding a positional representation $\mathbf{e}_i = \mathbf{e}_i^{\text{row}} + \mathbf{e}_i^{\text{col}} + \mathbf{e}_i^{\text{box}}$.

Reasoning Module. The reasoning module uses $L{=}6$ Transformer layers, $H{=}8$ attention heads, $d_{\text{model}}{=}256$, and $d_{\text{ff}}{=}4d_{\text{model}}$.

Hard Clue Restoration. At inference time, a final post-processing step replaces the Transformer output at evidence (clue) cells with the perception module's pre-reasoning logits, guaranteeing that clue-cell predictions exactly match the CNN's output. This architectural choice separates the evaluation of perception quality (measured at clue cells) from reasoning quality (measured at blank cells).

Declarative Constraint Specification. The complete constraint specification is shown in Fig. 2. The three integrity constraints induce 27 constraint groups (9 rows + 9 columns + 9 boxes), each generating 81 T_P residual terms, for a total of 2,187 differentiable constraint-satisfaction checks per forward pass, all derived automatically from the declarative specification.

```
:- R=1..9, V=1..9, #count{C : cell(R,C,V)} != 1.  % rows
:- C=1..9, V=1..9, #count{R : cell(R,C,V)} != 1.  % columns
:- BR=0..2, BC=0..2, V=1..9,
   #count{R,C : cell(R,C,V), R>=BR*3+1, R<=BR*3+3,
                             C>=BC*3+1, C<=BC*3+3} != 1.  % boxes
```

Figure 2: Complete declarative constraint specification for Visual Sudoku in Answer Set Programming (ASP) syntax. Each integrity constraint enforces that every digit appears exactly once in each row, column, and 3×3 box, respectively. These three rules define the mutual-exclusivity pattern used to construct the differentiable T_P residual loss manually.

Training Loss. The training loss combines four terms:

$$\mathcal{L} \;=\; \mathcal{L}_{\text{cnn}} + \mathcal{L}_{\text{sup}} + \lambda_{\text{blank}} \cdot \alpha(t) \cdot \mathcal{L}_{\text{blank}} + \lambda_{\text{ic}} \cdot \beta(t) \cdot \mathcal{L}_{\text{ic}}, \tag{3}$$

where $\mathcal{L}_{\text{cnn}}$ is cross-entropy on $\mathbf{z}^{\text{pre}}$ at evidence positions (trains the perception module directly), $\mathcal{L}_{\text{sup}}$ is cross-entropy on $\mathbf{z}^{\text{post}}$ at evidence positions (grounds the reasoning module at known anchors), $\mathcal{L}_{\text{blank}}$ is cross-entropy on $\mathbf{z}^{\text{post}}$ at latent positions against ground-truth labels (provides direct supervision where no clue is available), and $\mathcal{L}_{\text{ic}}$ is the T_P fixed-point loss (Eq. 2). Two curriculum schedules govern the strength of the latter two terms. $\alpha(t) = \max(0.1,\ 1 - 0.9\,t/T_{\text{decay}})$ decays latent-position cross-entropy from full weight to a $0.1\times$ residual floor over $T_{\text{decay}}{=}100$ epochs, preventing the model from fitting symbol identities at latent positions before it has learned to propagate constraint information. $\beta(t) = \min(1,\ t/T_{\text{warm}})$ linearly ramps the constraint loss from zero over $T_{\text{warm}}{=}20$ warmup epochs,

stabilizing early training before the T_P gradient becomes dominant. Hyperparameters are $\lambda_{\text{blank}} = 0.3$ and $\lambda_{\text{ic}} = 5.0$. The CNN backbone is frozen for the first 20 epochs, then unfrozen to allow constraint feedback to influence perception.

Inference-Time Constraint Enforcement. Although the T_P constraint loss trains the model to produce distributions approaching fixed points of the immediate-consequence operator, the argmax of the post-training softmax output is not guaranteed to satisfy all constraints exactly. Two complementary strategies close the remaining gap without invoking an external solver.

Iterative T_P Refinement. Before taking the argmax, the soft distributions $\mathbf{p}$ are refined by iterating the T_P operator K times. At each step, the row, column, and box T_P values are computed independently, averaged element-wise, and re-normalized to a valid probability distribution, as shown in Eq. (4),

$$\mathbf{p}_{i,s}^{(t+1)} \;=\; \frac{\frac{1}{|\mathcal{G}_i|}\sum_{G\ni i} T_{P,G}(\mathbf{p}^{(t)})_{i,s}}{\sum_{s'}\frac{1}{|\mathcal{G}_i|}\sum_{G\ni i} T_{P,G}(\mathbf{p}^{(t)})_{i,s'}}, \tag{4}$$

where $\mathcal{G}_i \subseteq \mathcal{G}$ denotes the set of constraint groups containing position i, $T_{P,G}$ denotes T_P applied within group G, and normalization is applied per-position. The iteration propagates exclusivity information from neighboring cells without discrete commitment and requires no gradient computation. On the Visual Sudoku test set, K=10 iterations achieve 98.7% constraint satisfaction, while 95.6% is the raw board accuracy.

Greedy Constrained Decoding. AS^2 also employs a final discrete decoding step via a confidence-ordered greedy decoding procedure that requires no external solver. Evidence cells (identified by the mask $\mathbf{m}$) are locked first, each assigned the argmax of the perception module's distribution. The assigned digit is then hard-zeroed from all cells sharing a constraint group with the locked cell, and the remaining distributions are renormalized. The procedure then iterates over latent cells in decreasing order of max-probability. At each step, the most confident unassigned cell is committed to its top remaining digit and its assignment is propagated to all peers by the same masking-and-renormalization step. The greedy decoder has no branching or backtracking and cannot revise an earlier commitment. If an earlier choice leads to a dead end, the procedure continues with its remaining scores, which can produce an incorrect assignment reflected in the final evaluation.

A.5 Additional CLEVR-Hans Results

Detailed Description. Formally, a CLEVR-Hans scene b contains $N_b \leq 10$ objects padded to a fixed maximum $N_{\max} = 10$ with a validity mask $\mathbf{m}_b \in \{0,1\}^{N_{\max}}$. Each object n is characterized by four categorical attributes drawn from the vocabulary

$$\mathcal{A} = \underbrace{\text{color}}_{8} \cup \underbrace{\text{shape}}_{4} \cup \underbrace{\text{size}}_{3} \cup \underbrace{\text{material}}_{2}, \quad |\mathcal{A}| = 17,$$

where the subscript denotes the number of values in each sub-domain, and the perception module produces per-object, per-attribute probability distributions $\mathbf{p}_b \in [0,1]^{N_{\max}\times 17}$. The seven concept classes of CLEVR-Hans7 are declared as conjunctive existential Horn rules and CLEVR-Hans3 uses only the first three. The ground-truth (non-confounded) rules are listed below, with spurious correlations present in training but absent at test time noted in parentheses.

CLEVR-Hans3 concept rules:

- C_0: $\exists$ large cube $\wedge$ $\exists$ large cylinder. (Confounded: large cubes are always gray in train/val.)
- C_1: $\exists$ small metal cube $\wedge$ $\exists$ small sphere. (Confounded: small spheres are always metal in train/val.)
- C_2: $\exists$ large blue sphere $\wedge$ $\exists$ small yellow sphere.

Additional CLEVR-Hans7 concept rules (C_3–C_6):

- C_3: $\exists$ small green object $\wedge$ $\exists$ small brown object $\wedge$ $\exists$ small purple object, with the scene containing at least five small objects overall.

- C_4: $\exists$ cyan object $\wedge$ $\exists$ red object $\wedge$ $\exists$ red object, where the cyan object is in front of the two red objects. (Confounded: the cyan object is always a small cube in train/val.)
- C_5: ($\exists$ sphere $\times 3$ in the left half of the scene) or ($\exists$ sphere $\times 3$ in the left half $\wedge$ $\exists$ metal cylinder $\times 3$ in the right half). Modeled by Eq. 8.
- C_6: $\exists$ large metal object $\wedge$ $\exists$ large metal object.

The presence of confounded classes (CLEVR-Hans3: C_0, C_1; CLEVR-Hans7: C_0, C_1, C_4) is a deliberate design choice of the benchmark [Stammer et al., 2021] to test whether a model generalizes to the true symbolic rule or overfits the spurious correlation present only during training. The explicit T_P operator hard-codes the ground-truth rules and is therefore immune to this confounder by construction. The learned T_P operator must recover the true rules from data, making its test-set performance a direct measure of symbolic generalization.

AS^2 Instantiation. The CLEVR-Hans instantiation of AS^2 specializes the generic framework along three dimensions, namely the perception front-end (which must localize individual objects in color 3-D scenes), the T_P operator (which uses the existential conjunctive instantiation described below rather than the mutual-exclusivity instantiation), and the training loss (which combines concept-level classification supervision with per-object attribute supervision). We evaluate four AS^2 variants per CLEVR-Hans benchmark by crossing two perception modules (Crop, a ResNet-18 with ROI-Align [He et al., 2017], and Scene, a ResNet-50 with bilinear feature-map sampling) with two T_P operators, namely Explicit, in which the ground-truth rules are hard-coded, and Learned, in which a weight matrix is initialized from the rules and refined end-to-end.

A.5.1 T_P Instantiation for Existential Conjunctive Rules

The mutual-exclusivity instantiation used for Visual Sudoku (Section 3) covers a large class of declarative constraints, but the same lifting recipe extends to qualitatively different rule families. CLEVR-Hans requires instantiating T_P for *existential conjunctive rules*, in which class membership is defined by the existence of entities that satisfy a conjunction of attribute conditions. The setting is as follows. A scene contains up to $N_{\max}$ entities, each described by a vector of D categorical attributes drawn from a finite vocabulary $\mathcal{A} = \bigcup_{a=1}^{D} \mathcal{A}_a$ of total size $|\mathcal{A}| = \sum_a |\mathcal{A}_a|$, with a binary validity mask $\mathbf{m} \in \{0,1\}^{N_{\max}}$ identifying genuine entities. Perception produces per-entity attribute distributions $\mathbf{p} \in [0,1]^{N_{\max} \times |\mathcal{A}|}$, replacing the per-position symbol distributions of the mutual-exclusivity instantiation. Each label class C_k ($k = 0, \ldots, K-1$) is declared as a conjunction of *required-entity templates* $\mathcal{R}_k = (r_{k,1}, \ldots, r_{k,m_k})$, where each $r_{k,j}$ specifies a conjunction of required attribute values that some entity in the scene must satisfy. The class assignment is true if and only if every template is matched by at least one valid entity.

Per-entity attribute satisfaction. Given $\mathbf{p}$ and template $r_{k,j}$ specifying a subset of (attribute, value) pairs, the soft satisfaction probability of entity n under $r_{k,j}$ is

$$s_{n,k,j} = \prod_{(a,v) \in r_{k,j}} p_{n,\, \mathrm{offset}(a) + \mathrm{idx}(a,v)}, \tag{5}$$

where $\mathrm{offset}(a)$ is the starting index of attribute a in the flat $|\mathcal{A}|$-dimensional vector and $\mathrm{idx}(a,v)$ is the position of value v within attribute a's sub-domain. Unspecified attributes contribute probability 1 to the product. This is the direct probabilistic analogue of testing whether entity n matches $r_{k,j}$ under a propositional interpretation.

Soft existential quantification. The existential condition "there exists an entity in the scene satisfying $r_{k,j}$" is approximated by a soft maximum over valid entity slots,

$$e_{k,j} = \max_{n \,:\, \mathbf{m}_n = 1} s_{n,k,j}. \tag{6}$$

This is a max-product aggregation, the standard differentiable approximation of the first-order existential quantifier under soft semantics [Badreddine et al., 2022].

Class satisfaction score. The T_P immediate consequence for class C_k combines the existential scores over all required templates via a product,

$$T_P(\mathbf{p})_k = \prod_{j=1}^{m_k} e_{k,j}, \tag{7}$$

and the per-class logit used for classification is computed in log-space as $z_k = \sum_j \log e_{k,j}$ for numerical stability. Unlike the mutual-exclusivity instantiation, this T_P operator is applied feed-forward without iterative fixed-point refinement. Class scores are computed in a single forward pass through Eqs. (5)–(7), and no residual loss of the form of Eq. (2) is applied.

Disjunctive classes. Some classes admit multiple alternative rule bodies, in which any one of several attribute configurations suffices to establish class membership. We handle this by computing a score per rule body and taking the soft maximum,

$$T_P(\mathbf{p})_k = \max_{\ell} \prod_{j=1}^{m_{k,\ell}} e_{k,\ell,j}, \tag{8}$$

where ℓ indexes the disjuncts. This handles multiple rule bodies for the same class head, the standard ASP treatment of disjunctive premises.

Learned operator variant. In addition to the explicit operator above, we introduce a *learned* variant that parametrizes the required-entity templates as a differentiable weight matrix $\mathbf{W} \in \mathbb{R}^{K \times R_{\max} \times |\mathcal{A}|}$, where $R_{\max}$ is the maximum number of required entities across all classes. The weight matrix is initialized from the explicit rules and refined end-to-end. Entity satisfaction scores become

$$s^{\text{learned}}_{n,k,j} = \big(\text{softmax}_A(\mathbf{W}_{k,j})\big)^\top \mathbf{p}_n, \tag{9}$$

and the log-sum-exp aggregation over entities uses a learnable inverse temperature $\tau > 0$,

$$e^{\text{learned}}_{k,j} = \frac{1}{\tau} \log \sum_{n=1}^{N_{\max}} \exp\big(\tau \cdot s^{\text{learned}}_{n,k,j}\big). \tag{10}$$

An entropy regularization term

$$\mathcal{L}_{\text{tp}} = 0.01 \, \frac{\sum_{b,n} \mathbf{m}_{b,n} \, H(\mathbf{p}_{b,n})}{\sum_{b,n} \mathbf{m}_{b,n}} \tag{11}$$

encourages peaked, interpretable attribute distributions, where $H(\cdot)$ denotes Shannon entropy. The explicit operator incurs no additional loss ($\mathcal{L}_{\text{tp}} = 0$).

Semantic character of explicit and learned operators. The explicit and learned operators differ in their semantic character. The explicit operator (Eq. 7) aggregates via a product over required templates, encoding a strict AND-conjunction in which every required template must be satisfied. The learned operator (Eqs. 9–10) replaces the product with a softmax-weighted dot product combined with a log-sum-exp over entities, encoding a soft mixture over candidate attribute configurations rather than a strict logical conjunction. This semantic divergence explains why the learned variant is more susceptible to confounder exploitation, since the soft mixture can absorb spurious attribute correlations in a way that the strict product cannot. When the ground-truth rules are known, the explicit operator is therefore preferred.

Object-Centric Perception. The CLEVR-Hans instantiation shares the reasoning module and attribute output heads with the Visual Sudoku variant but uses a different perception front-end that must localize individual objects from a 320×320 RGB scene image. We evaluate two complementary perception designs.

The *Crop Perception* design applies a shared ResNet-18 backbone [He et al., 2016] (pretrained on ImageNet) independently to a 64×64 crop centered on each object's ground-truth pixel coordinates, extracted via ROI-Align [He et al., 2017]. The 512-dimensional pool output is projected and layer-normalized to $\mathbb{R}^{d_{\text{model}}}$, providing strong bottom-up perception at the cost of missing global scene context.

The *Scene Attention Perception* design processes the full scene image with a ResNet-50 backbone to produce a $\lfloor H/32 \rfloor \times \lfloor W/32 \rfloor$ feature map. Per-object features are sampled from this map at each object's normalized pixel coordinates via bilinear interpolation (`grid_sample`), yielding a 2048-dimensional vector per object, projected to $\mathbb{R}^{d_{\text{model}}}$. This design encodes global context at the cost of reduced spatial precision for small objects.

Regardless of the perception front-end, the $N_{\max}$ object tokens (with padding positions masked out) are processed by an L=4-layer Transformer encoder with H=8 attention heads, d_{model}=256, and d_{ff}=$4d_{\text{model}}$, enabling the model to propagate evidence across objects before producing per-object attribute distributions. Four independent linear projection heads then produce logits for each attribute sub-domain:

$$\hat{\mathbf{c}}_n = W_c\,\mathbf{h}_n, \quad \hat{\mathbf{s}}_n = W_s\,\mathbf{h}_n, \quad \hat{\mathbf{z}}_n = W_z\,\mathbf{h}_n, \quad \hat{\mathbf{m}}_n = W_m\,\mathbf{h}_n,$$

where $W_c \in \mathbb{R}^{8\times d}$, $W_s \in \mathbb{R}^{4\times d}$, $W_z \in \mathbb{R}^{3\times d}$, and $W_m \in \mathbb{R}^{2\times d}$ correspond to color, shape, size, and material respectively. Softmax is applied per head to produce $\mathbf{p}_{b,n} \in [0,1]^{17}$, which is fed into the T_P operator (Eqs. 5–7). Padded object slots are zeroed out before the operator.

Training Loss. The CLEVR-Hans training loss combines concept-level classification supervision with per-object attribute supervision:

$$\mathcal{L} = \mathcal{L}_{\text{cls}} + \lambda_{\text{attr}}\,\mathcal{L}_{\text{attr}} + \mathcal{L}_{\text{tp}}, \tag{12}$$

where $\mathcal{L}_{\text{cls}}$ is cross-entropy on the concept logits $z_k = \log T_P(\mathbf{p})_k$ against the ground-truth class label, $\mathcal{L}_{\text{attr}}$ is the mean cross-entropy over the four per-object attribute heads at valid object slots against ground-truth attributes provided by the dataset, and $\mathcal{L}_{\text{tp}}$ is the entropy regularization term of Eq. (11), which is zero for the explicit operator and non-zero for the learned operator. The attribute loss weight is $\lambda_{\text{attr}} = 0.5$. No curriculum scheduling is applied, as attribute ground truth is available from the start and the explicit T_P operator requires no warm-up. We train for up to 100 epochs with AdamW ($\eta = 3 \times 10^{-4}$, $\lambda_{\text{wd}} = 10^{-2}$), a cosine learning-rate schedule with 100 linear warm-up steps, gradient clipping to norm 1.0, bf16 precision, and early stopping with patience 20 monitored on validation accuracy.

A.6 Additional Empirical Results: MNIST Addition

Introduced by Manhaeve et al. [2018] as the canonical benchmark for neuro-symbolic systems, $\text{MNIST}_{\text{Add}}$ presents a model with N MNIST digit images [LeCun et al., 1998] and requires it to predict both the individual digit labels and their sum. Correct inference jointly requires (i) accurate digit recognition and (ii) adherence to arithmetic constraints. We evaluate three scales: N=2 (sum range $[0, 18]$, 19 classes), N=4 (sum range $[0, 36]$, 37 classes), and N=8 (sum range $[0, 72]$, 73 classes), each with 30,000 training and 5,000 test pairs. This benchmark is well-saturated in the literature, meaning most methods achieve near-perfect accuracy. We include it to confirm that AS^2 is competitive on a standard arithmetic-constraint task and to study scaling behavior as N increases. Table 6 compares digit accuracy for our two-digit model, trained with both digit and sum labels, with results for methods trained using only sum labels, as reported by Takemura and Inoue [2024].

Table 6: MNIST Addition digit accuracy (%). Prior results are from Takemura and Inoue [2024], Table 2 (Addition 1). DPL, DSL, and NASP denote DeepProbLog, DeepStochLog, and NeurASP. I and C denote their implication-only and constraint-only variants.

Method	Digit accuracy (%)
DPL [Manhaeve et al., 2018]	97.8
DSL [Winters et al., 2022]	95.8
NASP [Yang et al., 2020]	97.7
I [Takemura and Inoue, 2024]	97.7
C [Takemura and Inoue, 2024]	97.5
I+C [Takemura and Inoue, 2024]	97.4
AS^2, $N = 2$	99.73 ± 0.03

Baselines. The following systems report results on $\text{MNIST}_{\text{Add}}$ and are included in Table 7.

- DeepProbLog [Manhaeve et al., 2018, 2021a] integrates neural networks into ProbLog with exact gradients via weighted model counting. Results taken from van Krieken et al. [2023].
- NeurASP [Yang et al., 2020] wraps neural outputs as probabilistic facts in an ASP program. Result at $N{=}2$ taken from Manhaeve et al. [2021b].
- Scallop [Li et al., 2023] implements differentiable Datalog with provenance semirings. Best variant reported by Li et al. [2023].
- A-NeSI [van Krieken et al., 2023] provides approximate scalable inference for probabilistic neuro-symbolic programs. Results taken from van Krieken et al. [2023].

All four systems use sum-level supervision only (no per-digit labels). AS^2 uses both per-digit and sum supervision (Eq. 13), so Table 7 should be read as evidence of architectural competitiveness on a saturated benchmark rather than a direct head-to-head comparison.

Arithmetic T_P Operator. The MNIST Addition task introduces a single arithmetic global-sum constraint that does not fit the mutual-exclusivity template used for Visual Sudoku. Let $\mathbf{p} \in [0,1]^{N\times 10}$ denote the per-digit distributions produced by perception. The probability that the sum of N digits equals $t \in \{0,\dots,9N\}$ is the polynomial convolution $q_t = (\mathbf{p}_1 * \cdots * \mathbf{p}_N)_t$, where $\mathbf{p}_n = (p_{n,0},\dots,p_{n,9})$. A leave-one-out variant of this convolution defines, for every position $i \in \{1,\dots,N\}$ and every digit $s \in \{0,\dots,9\}$, the probability that the remaining $N{-}1$ positions sum to the residual $t-s$, namely $T_P(\mathbf{p})_{i,s,t} = p_{i,s}\,(*_{n\neq i}\mathbf{p}_n)_{t-s}$. Marginalizing over the ground-truth target sum $t^\star$ yields the sum-conditioned digit distribution that the learned digit head is trained to match. The total training loss combines digit supervision, sum supervision, and the arithmetic T_P residual,

$$\mathcal{L}_{\text{mnist}} \;=\; \mathcal{L}_{\text{digit}} + \lambda_{\text{sum}}\,\mathcal{L}_{\text{sum}} + \lambda_{\text{tp}}\,\big\|\mathbf{p} - T_P(\mathbf{p} \mid t^\star)\big\|_2^2, \tag{13}$$

where $\mathcal{L}_{\text{digit}}$ is per-digit cross-entropy, $\mathcal{L}_{\text{sum}}$ is cross-entropy on the predicted sum distribution $\mathbf{q}$, and the third term anchors the learned distributions to a fixed point of the leave-one-out arithmetic operator under the observed sum. We use $\lambda_{\text{sum}} = 1.0$ and $\lambda_{\text{tp}} = 0.1$ for all reported MNIST Addition runs.

Metrics. We report Digit-Acc (fraction of individual digits predicted correctly) and Sum-Acc (fraction of instances where the predicted sum is correct). Because the baselines do not produce per-digit predictions, Digit-Acc is listed as N/A for them.

Table 7: Results on $\text{MNIST}_{\text{Add}}$. Digit-Acc = individual digit accuracy (%); Sum-Acc = sum accuracy (%). Results for prior work are taken from the respective publications. **Bold** = best in column. "†" indicates the result uses only sum-level supervision; Digit-Acc is not reported for these methods because per-digit predictions are implicit. All AS^2 and ablation results use both digit and sum supervision.

	$N{=}2$		$N{=}4$		$N{=}8$	
Method	Digit-Acc ↑	Sum-Acc ↑	Digit-Acc ↑	Sum-Acc ↑	Digit-Acc ↑	Sum-Acc ↑
DeepProbLog † [Manhaeve et al., 2021a]	N/A	97.20±0.50	N/A	N/A	N/A	N/A
NeurASP † [Yang et al., 2020]	N/A	97.30±0.30	N/A	N/A	N/A	N/A
Scallop † [Li et al., 2023]	N/A	98.22	N/A	97.00	N/A	N/A
A-NeSI † [van Krieken et al., 2023]	N/A	97.66±0.21	N/A	95.96±0.38	N/A	N/A
Perception-Only	99.69±0.05	99.37±0.10	99.91	99.64	**99.98**	**99.86**±0.03
Pipeline	99.63±0.01	99.26±0.02	99.89±0.03	99.57±0.13	**99.98**±0.00	99.89±0.03
AS^2 (ours)	**99.73**±0.03	**99.45**±0.08	99.87	99.44	99.95±0.01	99.01±0.18

Results. At $N{=}2$, AS^2 achieves 99.73±0.03% digit accuracy and 99.45±0.08% sum accuracy. As N increases to 4 and 8, the benchmark becomes saturated for all three model variants (all above 99.8% digit accuracy). At $N{=}8$, the Perception-Only and Pipeline baselines achieve slightly higher sum accuracy than AS^2 (99.86% and 99.89% vs. 99.01±0.18%). This gap is expected: the arithmetic T_P operator requires leave-one-out convolutions over 8 addends, and the constraint loss gradient introduces optimization pressure that competes with direct digit and sum supervision. Nevertheless, digit accuracy of AS^2 remains above 99.9% at all scales, confirming that the soft constraint mechanism does not degrade perception quality. Because the benchmark is saturated, we treat it as a sanity check

that the architecture generalizes to arithmetic constraints and reserve detailed empirical analysis for the harder Visual Sudoku and CLEVR-Hans benchmarks in the main text.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract states three concrete claims: (i) full end-to-end differentiability through a soft T_P operator, (ii) replacement of conventional positional embeddings by constraint-group membership embeddings on spatial tasks, and (iii) specific accuracy figures on Visual Sudoku and CLEVR-Hans. All three claims are directly supported by the experimental results in Section 4 and the theoretical derivation in Section 3.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 5 includes a dedicated Limitations paragraph. No theoretical guarantee exists that the learned soft distributions will prevent dead ends on problems that require backtracking.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: Section 3 defines the probabilistic T_P operator and derives the fixed-point residual loss $\mathcal{L}_{\text{ic}}$ (Eq. 2). The derivation establishes that the uniform distribution is not a minimizer of $\mathcal{L}_{\text{ic}}$, distinguishing it from naive penalty formulations, and that any valid one-hot assignment satisfying all constraints yields $\mathcal{L}_{\text{ic}} = 0$. The construction is grounded in the classical least-fixed-point semantics of the immediate-consequence operator.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 4 and the Supplementary Material provide all necessary details for reproducing the main experimental results, including architecture specifications, training protocols, and evaluation procedures.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Code is publicly available at `https://huggingface.co/wabdalm/AS2`.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Architecture dimensions and evaluation protocols are provided in Section A.4, Section A.5, and Section A.6. Perception backbones (a three-block CNN for Visual Sudoku; ResNet-18 with ROI-Align or ResNet-50 with bilinear sampling for CLEVR-Hans) are described in the respective appendix sections. Declarative constraint specifications (Clingo ASP programs and relevant differentiable operators) are described in Section 3 and Section A.4.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: CLEVR-Hans results for AS^2 are reported as mean $\pm$ standard deviation over three seeds in Table 2, while the baseline statistics from Stammer et al. [2021] use five seeds. Visual Sudoku results are reported on a fixed evaluation set without multi-seed variance. MNIST Addition results in Table 7 include $\pm$ values.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Section A.2 reports that all experiments were run on a single NVIDIA H200 (143 GB HBM3). Training time is approximately 2–4 hours for Visual Sudoku and CLEVR-Hans, and under 1 hour per scale for MNIST Addition.

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: This work presents a neuro-symbolic reasoning architecture evaluated on standard perception benchmarks. It does not involve human subjects, sensitive data, or dual-use concerns. All experimental assets use publicly licensed datasets and open-source libraries.

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Section 5 contains a Broader Impact paragraph discussing (i) positive impacts, including potential applications in material sciences and drug discovery, and (ii) potential misuse if the framework is applied to automate consequential decisions, such as medical diagnosis, without adequate human oversight.

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The released assets (model code, trained checkpoints on public benchmarks) present no elevated dual-use risk. No personal data, private datasets, or high-risk generative capabilities are involved.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: MNIST is loaded through `torchvision`, and the CLEVR-Hans distribution is credited to Stammer et al. [2021]. Visual Sudoku boards are generated and rendered with MNIST digits. The `torchvision` software uses the BSD-3-Clause license (`https://github.com/pytorch/vision/blob/main/LICENSE`), while the terms for the pretrained ResNet-18 ImageNet weights require separate verification. Licenses and version-specific terms for all datasets and pretrained checkpoints are not fully documented.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [No]

Justification: The AS^2 software package constitutes the primary new asset. No new datasets are introduced.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: No crowdsourcing or human-subject experiments were conducted.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: No human subjects are involved in this research.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLMs were used only for editing, formatting, and basic coding assistance, not as a component of the core research method.