Tensor Logic: The Language of AI

A common index language for rules, neural computation, and factors

Paper by Pedro Domingos

arXiv:2510.12269v3 26 October 2026

The main idea is a shared computational grammar

One-sentence takeaway

A Datalog rule can be evaluated as a Boolean tensor contraction followed by a threshold; the same index grammar also describes numerical AI models.

$$\text{join} \longrightarrow \text{sum over hidden indices} \longrightarrow \text{optional nonlinearity}$$

- The paper supplies language semantics and illustrative model encodings.
- Its runtime and embedding-reliability claims are proposals to test.

Domingos, Secs. 1 and 3–6, PDF pp. 1–16.

The desired language spans three existing strengths

Logic programming

Facts, rules, and sound deduction for the Boolean fragment.

Tensor frameworks

Differentiation and regular dense computation on accelerators.

Graphical models

Factor products and marginalization over variables.

Goal: express all three with indexed equations.

Paper, Sec. 1, PDF pp. 1–2.

A Datalog rule is a join followed by projection

$$\text{Aunt}(x, z) \leftarrow \text{Sister}(x, y), \text{Parent}(y, z)$$

1. Join facts that agree on the shared variable y.
2. Project away y: keep only (x, z).
3. A fact is true if at least one witness y exists.

Teaching example

Sister(Alice, Bob) and Parent(Bob, Charlie) derive Aunt(Alice, Charlie).

Paper, Secs. 2.1 and 3.1, PDF pp. 2–5. Example instantiation for discussion.

Boolean contraction implements the rule

$$A_{xz} = H\left(\sum_{y} S_{xy} P_{yz}\right), \qquad H(t) = \begin{cases} 1 & t > 0 \\ 0 & t \leq 0 \end{cases}$$

- Multiplication enforces simultaneous facts for one y.
- Summation enumerates witnesses; H turns any positive count into truth.
- The Boolean inputs and threshold are essential to the logical reading.

Paper, Sec. 3.1, PDF p. 5.

Join and projection are the language primitives

$$(U \bowtie V)_{\alpha\beta\gamma} = U_{\alpha\beta} V_{\beta\gamma}, \qquad \pi_\alpha(T) = \sum_\beta T_{\alpha\beta}.$$

- Join retains shared indices β; projection removes indices absent from the left-hand side.
- No shared index gives a tensor product; all shared indices give pointwise multiplication.
- For Boolean relations, projection plus threshold recovers database projection.

Paper, Sec. 3.1, PDF pp. 5–6.

A program is a set of indexed equations

```
Y = step(W[i] X[i])     Y[i] = step(W[i] X[i])
```

- An index absent from the left-hand side is summed out; a retained index is not.
- Missing tensor entries default to zero; equations with the same output are summed.
- Boolean facts use Datalog-style parentheses; numerical tensors use brackets.

Consequence

One syntax can encode both sparse facts and dense numerical transformations.

Paper, Sec. 3.1, PDF pp. 5–7.

Inference can be data driven or query driven

Forward chaining

Execute applicable equations repeatedly; stop when no new elements appear or a criterion is met.

Backward chaining

Start from a query; recursively evaluate only its required input tensor elements.

Exact Datalog soundness applies to the Boolean rule fragment.

Paper, Secs. 2.1 and 3.2, PDF pp. 3, 7.

Differentiation preserves the index structure

$$Y_{ij} = \sum_k M_{ik} X_{kj} \quad \Rightarrow \quad \frac{\partial L}{\partial M_{ik}} = \sum_j \frac{\partial L}{\partial Y_{ij}} X_{kj}.$$

- The gradient is another join and projection of tensors.
- Losses are equations; parameters are tensors not supplied as data.
- Example-specific derivations motivate backpropagation through structure.

Paper, Sec. 3.3, PDF pp. 7–8. Full loss-gradient equation expanded for teaching.

The paper encodes four AI paradigms

Paradigm	Core correspondence
Neural models	Indexed linear maps, aggregations, activations
Symbolic AI	Datalog facts and rules as Boolean equations
Kernel machines	Kernel values join support-vector weights
Graphical models	Factors as tensors; marginals as projections

These are expression examples, not experimental comparisons.

Paper, Sec. 4, Tables 1–3, PDF pp. 8–13.

A GNN is indexed neighborhood aggregation

$$\mathrm{Agg}_{nld} = \sum_{n'} \mathrm{Neig}_{nn'} Z_{n'ld}.$$

- $\mathrm{Neig}_{nn'}$ is the graph relation; Z is a transformed node feature.
- Join selects neighbors and projection sums their messages.
- A later equation combines neighbor and self representations.

Paper, Sec. 4.1 and Table 1, PDF pp. 9–10.

Attention becomes a sequence of contractions

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V,$$

$$C_{pp'} = \mathrm{softmax}_{p'}\left(\frac{\sum_d Q_{pd} K_{p'd}}{\sqrt{D_k}}\right), \qquad \mathrm{Attn}_{pd_v} = \sum_{p'} C_{pp'} V_{p'd_v}.$$

- Position, block, head, and feature are explicit tensor indices.
- Table 2 sketches embedding, attention, residual flow, and output.

Paper, Sec. 4.1 and Table 2, PDF pp. 10–11. Formula restated from the prose; table syntax is schematic.

Symbolic rules and kernel machines fit the same grammar

Symbolic AI

Datalog rules are Boolean tensor equations. Function-free reasoning and planning inherit the rule representation.

Kernel machine

$Y(Q) = f(\sum_i a_i y_i K(Q, i) + b)$ joins each support vector with its kernel similarity.

Prolog-style functions would need a unification mechanism.

Paper, Secs. 4.2–4.3, PDF p. 11.

Graphical-model operations map directly

$$P(x) = \frac{1}{Z} \prod_k \phi_k(x_{\{k\}}), \qquad \phi'(X) = \sum_Y \phi(X, Y).$$

- Factor $\leftrightarrow$ nonnegative tensor; factor product $\leftrightarrow$ join.
- Marginalization $\leftrightarrow$ projection; conditioning needs normalization.
- Join-tree inference is linear in *tree size*, which may be exponential in the original graph.

Paper, Sec. 4.4 and Table 3, PDF pp. 12–13.

Random embeddings encode sets approximately

$$s = \sum_{x \in S} e_x, \qquad \text{score}(a) = s^\top e_a.$$

- With random unit vectors, members score near 1 and nonmembers near 0.
- Paper's noise scale: $\sqrt{N/D}$ for set size N, embedding dimension D.
- Threshold $1/2$ trades memory for a nonzero probability of error.

Paper, Sec. 5, PDF p. 13.

Outer-product superposition encodes a relation

$$E_R = \sum_{(x,y)\in R} e_x \otimes e_y, \qquad \widehat{R}(a,b) = e_a^\top E_R e_b.$$

- A tuple contributes an outer product of its two object vectors.
- Querying contracts the stored tensor with both argument vectors.
- Other tuples create cross-term noise when embeddings are not orthogonal.

Paper, Sec. 5, PDF pp. 13–14.

Rules can operate on embedded relations

Author's proposal

Embed rule antecedents and consequents; run forward or backward chaining over the resulting tensor equations.

- Near-orthogonal random vectors make extracted facts approximately correct.
- The paper suggests periodic extraction, thresholding, and re-embedding to limit error.
- It gives no end-to-end error bound across long derivations.

Paper, Sec. 5, PDF pp. 14–15. Final point is our assessment.

Learned embeddings add analogical transfer

$$G_{xx'} = e_x^\top e_{x'}.$$

- G is a Gram matrix of object similarities.
- Similar objects can "borrow" one another's inferences.
- This is continuous generalization, distinct from exact Boolean deduction.

What must be controlled

Similarity leakage, thresholding, and error accumulation during chaining.

Paper, Sec. 5, PDF pp. 14–15.

Zero temperature does not imply exact deduction

$$e_a = (1, 0), \quad e_b = (0.8, 0.6), \quad G_{ab} = 0.8,$$

$$\sigma(G_{ab}, T) = \frac{1}{1 + e^{-0.8/T}} \xrightarrow[T \to 0^+]{} 1.$$

- Both are unit vectors, yet the off-diagonal match survives.
- Taking $T \to 0$ thresholds similarity signs; it does not make arbitrary G the identity.
- Exact deduction therefore needs additional matching constraints or discrete recovery.

Paper claim: Sec. 5, PDF p. 15. Counterexample and inference: this discussion.

Scaling has two proposed implementation routes

Sparse + dense split

Database engine handles sparse joins; GPUs handle dense subtensor operations.

Compressed dense route

Embed or decompose sparse relations, then evaluate dense contractions on GPUs.

Cost depends on sparsity, representation rank, target error, and query structure.

Paper, Sec. 6, PDF pp. 15–16. Cost conditions are our assessment.

The evidence is expressive, not empirical

Claim	Evidence in the paper
Shared rule/tensor syntax	Definitions and worked equations
Model expressibility	GNN, transformer, kernel, and factor examples
Fast large-scale execution	Two implementation proposals; no benchmark
Embedding soundness	No proof; low-T counterexample

Paper, Secs. 3–6, PDF pp. 4–16; final assessment is ours.

Takeaways for a research agenda

1. The join–projection–threshold bridge is the paper's strongest concrete insight.
2. A useful language needs executable semantics, sparse/dense planning, and differentiation rules.
3. Embedding-space deduction requires explicit correctness conditions and measured error growth.

Discussion question

Which operations should stay exact, and where is approximate analogical transfer worth its cost?

Paper, Secs. 3–7, PDF pp. 4–16; research priorities are our synthesis.