TensorLang

v0.3 — Reference Specification

An experimental tensor-equation language for symbolic, numeric, relational, and dynamical AI models.

This document mirrors the on-page reference at the TensorLang website and is intended as a portable companion for engineers, researchers, and tool builders evaluating the language.

ATTRIBUTION

Inspired by Pedro Domingos, "Tensor Logic: The Language of AI", arXiv:2510.12269 — arxiv.org/pdf/2510.12269.

TensorLang is an independent experimental project. It is **not** Tensor Logic itself, and it is not affiliated with the author or the University of Washington.

Contents

Inspired by Tensor Logic

TensorLang begins from the ideas in Pedro Domingos's paper, which proposes tensor equations as a unifying language for neural, symbolic, and statistical AI. TensorLang is an independent experimental implementation path inspired by that work. It is not Tensor Logic itself, and should not be presented as an official implementation or affiliated project.

Field	Value
Author	Pedro Domingos
Institution	Paul G. Allen School of Computer Science & Engineering, University of Washington
arXiv	2510.12269
Link	https://arxiv.org/pdf/2510.12269

Disclosure

- TensorLang is **not** Tensor Logic.
- No affiliation with Pedro Domingos or the University of Washington.
- An independent practical implementation/research language inspired by the paper's core idea that tensor equations can bridge symbolic logic, neural computation, and statistical AI.
- Presented as a research prototype exploring ideas motivated by the paper, not as the original invention of tensor logic.

2 — OVERVIEW

What is TensorLang?

TensorLang is a small experimental language for writing tensor equations over finite domains, relations, numeric arrays, semirings, fixed points, and dynamical systems. It is designed as a reference core for exploring neural-symbolic reasoning, finite Boolean verification, relational algebra lowering, and time-aware model simulation.

Three pillars

- **Tensor-first.** Programs are built from typed tensors, shapes, domains, equations, einsum, semirings, and fixed points.
- **Reasoning-aware.** Finite Boolean programs can declare properties and be checked by exhaustive finite-model verification, with SMT and Lean export hooks.
- **Dynamical.** TensorLang can describe simple continuous-time and discrete-time systems for software simulation of time-aware models.

3 — MOTIVATION

Why tensor equations?

Most AI stacks split neural, symbolic, relational, probabilistic, and dynamical systems across separate frameworks. Tensor equations are a candidate substrate that can describe all of them with shared structure.

- Tensors naturally represent neural computation.
- Boolean tensors can represent relations.
- Einsum can represent joins and projections.
- Semirings let the same structural operation mean different things depending on algebra.
- Fixed points model closure, reachability, ancestry, policy expansion, and recursive relations.
- Dynamic equations model evolving state over time.

Traditional split vs. TensorLang direction

Traditional split	TensorLang direction
Neural models	Common tensor-equation representation
Symbolic rules	Explicit domains and shapes
Relational queries	Semiring-aware operations
Probabilistic scoring	Verification hooks
Dynamical systems	Simulation and tracing

A small S-expression core

TensorLang programs are S-expressions describing modules, domains, typed tensors, semirings, equations, fixed points, properties, and dynamical systems. The surface is intentionally minimal so that semantics and tooling stay legible.

examples/family.tl — finite domains and tensors

```
(module examples.family
  (tensorlang "0.3")

  (domain Person
    (size 4)
    (labels "alice" "bob" "carol" "dave"))

  (tensor parent
    (type Bool)
    (shape Person Person)
    (data
      (row 0 1 0 0)
      (row 0 0 1 0)
      (row 0 0 0 1)
      (row 0 0 0 0))))
```

examples/ancestor.tl — fixed-point relation

```
(module examples.ancestor
  (import examples.family)

  (semiring OrAnd
    (plus or) (times and)
    (zero false) (one true))

  (tensor ancestor
    (type Bool)
    (shape Person Person)
    (semiring OrAnd))

  (define ancestor
    (fixpoint
      (or parent
          (einsum "ik,kj->ij" ancestor parent)))))
```

examples/reservoir.tl — discrete dynamical system

```
(module examples.dynamics
  (domain Node (size 32))

  (tensor W   (type Real) (shape Node Node))
  (tensor x0  (type Real) (shape Node))

  (system reservoir
    (state x (shape Node) (init x0))
    (step  x' (= (tanh (einsum "ij,j->i" W x))))
    (simulate
      (mode discrete)
      (steps 200)
      (trace x))))
```

Implementation status

TensorLang is an experimental reference implementation. The point of this section is to be transparent about what exists today versus what is staged research.

Implemented now

- S-expression source syntax
- Domains
- Typed tensors
- Tensor definitions
- Static shape and dtype checking
- Dependency graph and cycle detection
- Explicit semiring declarations
- Semiring-aware einsum
- Monotone finite Bool fixed-point checking
- Finite Bool property checking
- Exhaustive finite-model verification
- SMT-LIB export hooks
- Lean theorem-obligation export hooks
- Semiring-aware optimization
- Relational algebra lowering
- Optional numeric backend adapters
- Minimal trainable subset
- Capsulang-facing reasoner manifests
- Dynamic system declarations
- Continuous-time and discrete-time simulation
- Euler integration
- Reservoir-style recurrent examples
- Trajectory trace output

Explicitly not (yet)

- A replacement for PyTorch, JAX, or TensorFlow
- A production GPU compiler
- A full implementation of Pedro Domingos's Tensor Logic
- A full proof system
- A physical analog / neuromorphic hardware runtime
- A new foundation model

Short reference programs

Shortest paths via min-plus semiring

```
(semiring MinPlus
  (plus min) (times +)
  (zero +inf) (one 0))

(tensor dist
  (type Real) (shape Node Node)
  (semiring MinPlus))

(define dist
  (fixpoint
    (min edge
         (einsum "ik,kj->ij" dist edge))))
```

Property declaration and verification

```
(property reflexive
  (forall (i Person)
    (= (ancestor i i) false)))

(verify reflexive
  (mode exhaustive)
  (export smt  "out/reflexive.smt2")
  (export lean "out/reflexive.lean"))
```

7 — STATIC SEMANTICS

What the compiler checks before you run

TensorLang's front-end performs structural and algebraic checks ahead of any backend lowering. The intent is to catch shape, dtype, semiring, and recursion mistakes statically.

Check	Description
shape	Static rank and dimension checks per equation
dtype	Boolean / integer / real propagation through einsum
domain	Index sets are first-class, referenced by name
semiring	Each tensor binds to a declared algebraic structure
dependency	Definitions form a DAG; cycles allowed only via fixpoint
monotonicity	Fixed points checked as monotone over the chosen lattice

8 — VERIFICATION

A small, honest verification ladder

Verification today is finite-model exhaustive checking with hooks to export obligations. It is not a full proof system, and it is not a substitute for SMT solvers or proof assistants — it is a bridge into them.

Step	Stage	Description
1	Declare	Write a property over Boolean tensors and finite domains.
2	Enumerate	Exhaustively expand over the finite model.
3	Check	Evaluate the property against every assignment.
4	Export	Emit SMT-LIB or Lean obligations for external proof tools.

9 — ALGEBRA

Semirings and relational algebra

The same einsum expression can mean matrix multiplication, a join, a shortest path, or a max-product computation, depending on the declared semiring.

Semiring (+, x, 0, 1)	Carrier	Reading
(+, x, 0, 1)	Real	standard matrix multiplication / neural layers
(or, and, false, true)	Bool	relational join, transitive closure
(min, +, +inf, 0)	Real	shortest paths
(max, +, -inf, 0)	Real	Viterbi / longest path
(max, x, 0, 1)	Real	max-product inference

10 — DYNAMICS LAYER

Continuous- and discrete-time systems

A system declares state tensors, transition equations, and a simulation mode. Discrete-time steps and Euler integration for continuous-time are supported today, with reservoir-style recurrent examples included.

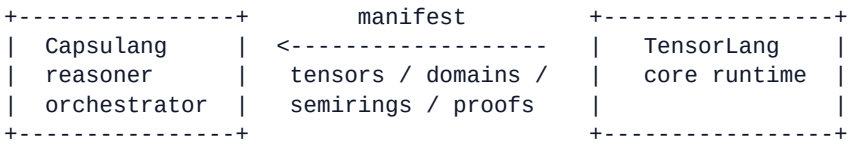
Reservoir-style recurrent system

```
(module examples.dynamics
  (domain Node (size 32))

  (tensor W   (type Real) (shape Node Node))
  (tensor x0  (type Real) (shape Node))

  (system reservoir
    (state x (shape Node) (init x0))
    (step  x' (= (tanh (einsum "ij,j->i" W x))))
    (simulate
      (mode discrete)
      (steps 200)
      (trace x))))
```

11 — INTEROP

Relationship to Capsulang

TensorLang exposes reasoner manifests intended for use as the tensor-equation core inside larger reasoning environments such as Capsulang. The boundary is intentionally narrow: TensorLang owns tensors, semirings, fixpoints, and verification obligations.



12 — DIRECTION

Roadmap

Phase	Items
v0.3 — now	Static semantics; Finite verification; Semiring einsum; Discrete + Euler dynamics
v0.4	Richer relational lowering; Improved SMT export; Differentiable subset expansion
v0.5	Probabilistic semirings; Better Lean obligation shape; Backend adapters
research	Symbolic <-> neural bridges; Larger trainable fragments; Hardware-aware lowering

Download & toolchain

The reference implementation ships as a small CLI and a library. Programs are .tl S-expression files; the toolchain provides parsing, type checking, semiring lowering, finite verification, and simulation.

Quick start

```
; 1. unzip the examples
unzip tensorlang-examples-v0.3.zip
cd tensorlang-examples

; 2. static check (syntax + shape + dtype)
tl check  family.tl

; 3. verify a finite property
tl verify verify_demo.tl --backend smt

; 4. simulate a dynamical system
tl run    reservoir.tl --trace h --steps 64

; 5. export to a host runtime
tl export shortest_path.tl --to numpy > shortest_path.py
tl export ancestor.tl      --to lean  > ancestor.lean
```

TensorLang v0.3 reference specification. Independent research prototype inspired by Pedro Domingos, "Tensor Logic: The Language of AI" (arXiv:2510.12269). Not affiliated with the author or the University of Washington.