From Triangles to Baseball to Quantum: A Data-First Path to Big Truths

No stepwise code. No “update” calls. Just enumerating five primitives to make logic emerge.

Introduction

Most of us treat these as separate wonders:

1. The Pythagorean theorem in geometry,
2. Baseball scoring, and
3. Quantum wavefunction “collapse.”

They seem unrelated—one is purely mathematical, one is a sports procedure, and one is advanced physics. But here’s the twist: *you can derive all three* using precisely the **same approach**: enumerating a small set of factual statements plus constraints—no special “update” functions required. By the time you list the relevant facts (edges/angles, runs/outs, amplitudes/measurements), each domain “snaps” into place. A “right triangle” can’t help but satisfy a2+b2=c2a^2 + b^2 = c^2a2+b2=c2, a baseball game can’t help but tally runs at three outs, and a quantum system can’t help but collapse when a measurement is declared.

> *Important note*: We aren’t saying these domains are “literally the same.” We’re just showing how a unified, declarative approach can capture all three without procedural illusions.

There’s a bigger story, one we won’t fully reveal until Part 4: a universal principle called the **Conceptual Model Completeness Conjecture** (CMCC). It basically says: “Everything from geometry to quantum can be expressed by enumerating facts and aggregator relationships.” Let’s see how that shapes up, domain by domain.

Part 1: Triangles

A handful of facts about edges and angles force out Pythagoras.

We all learn a2+b2=c2a^2 + b^2 = c^2a2+b2=c2 for right triangles. Usually, it’s taught via a standard geometric proof or an algebraic rearrangement of squares. But there’s a radical alternative: just list certain constraints about angles and edges, and realize you *cannot* form a right triangle that dodges Pythagoras.

The Verbatim Facts

1. A **Polygon** is a closed loop of 3 or more edges and a `count_of_edges`.
2. **Edges** have a `length` and a `squared_length`.

3. If a polygon's `count_of_edges` is 3, it is a **triangle**.

4. A square has 4 edges (just another fact, not our main focus).

5. Every polygon has interior angles.

6. If a shape's angles do not sum to 180°, it's *not* a triangle.

7. A **right triangle** is a triangle that has one 90° angle.

8. The longest edge in a right triangle is the **hypotenuse**.

9. The sum of the `squared_length` of the other two edges can be computed.

10. **pythagorean_theorem_fails**: if a right triangle's hypotenuse length squared ≠ the sum of the other edges' squared lengths, it fails.

At a glance, you see how the theorem arises. Steps 1–7 define "triangle" and highlight the 90° angle. Then steps 8–10 box you in: if you claim something is a "right triangle," you can't simultaneously let c2≠a2+b2c^2 \neq a^2 + b^2c2=a2+b2. The model flags that as contradictory. No separate "theorem function" or geometry proof is needed. By enumerating these facts, geometry itself refuses to let you store a right triangle that breaks Pythagoras.

Part 2: Baseball

Runs, Outs, and a Snapshot-Consistent "Scoreboard."

In geometry, enumerating edges and angles forced a2+b2=c2a^2 + b^2 = c^2a2+b2=c2. In baseball, enumerating **RunEvents** and **OutEvents** forces a scoreboard. Typically, you might imagine:

cpp

Copy

```cpp
if (a_run_happens) {

   runs_for_teamA += 1;

}

if (outs >= 3) {

   endCurrentInning();

}
```

But the purely **declarative** approach is simpler:

- **RunEvent**: “Run in the 3rd inning”
- **OutEvent**: “Out in the bottom of the 5th inning”

Aggregator constraints then tally how many runs or outs occur in each half-inning. If `outs ≥ 3`, that half-inning is “closed.” Instead of calling `endCurrentInning()`, we just have a constraint: `inning_over = (outs ≥ 3)`. Once the fact “3rd out” is declared, no partial or contradictory scoreboard can exist—it’s *automatically* a closed half-inning.

Scaling Up to Stats or Different Leagues

- **Advanced Metrics** (WAR, ERA, wOBA, etc.): all just sums and ratios of “events.”
- **Different Rules** (T-ball vs. MLB vs. Japanese baseball): let each `GameType` define how many outs, how many innings, etc.

The concept is the same as the triangle example: once we say “3 outs,” you can’t have a continuing half-inning. The scoreboard is never “out of sync” because everything emerges from enumerated facts plus aggregator logic—the “rules” that unify the final state.

Part 3: Quantum Mechanics

Wavefunction “collapse” or superposition, minus the mystical steps.

Now for something that might seem *even more* procedural: quantum measurement. Commonly, we see code like `collapseWavefunction()`. But in this data-based approach, you simply store amplitude values (like `a`, `b`) for each quantum state, then declare a **MeasurementEvent** that says “we observed spin-up.” Your aggregator constraints handle normalization and exclusive outcomes.

- **Wavefunction**: A record with fields for `amplitude_0` and `amplitude_1` (if it’s a single qubit).
- **MeasurementEvent**: “At time T, outcome = spin-up from wavefn #12.”
- **Aggregator Constraints**:
    1. Probability sums must be 1.
    2. If spin-up was observed, `a → 1` and `b → 0` in that snapshot (Copenhagen style).
    3. Storing “spin-up = 100%” and “spin-down = 100%” simultaneously is as contradictory as a “right triangle” with c2≠a2+b2c^2 \neq a^2 + b^2c2=a2+b2.

That's all "collapse" is here: a forced outcome of enumerating facts (wavefunction states, measurement data). Exactly as enumerating edges forced the Pythagorean theorem, and enumerating run events forced the scoreboard.

> Curious about multi-qubit or double-slit experiments? The same principle applies—you just enumerate additional amplitude values, constraints, and measurement events. Our GitHub repository shows how to replicate classic interference patterns in a purely data-first way.

Part 4: The Conjecture & Conclusion

The "Conceptual Model Completeness Conjecture" (CMCC)

Here is where all the puzzle pieces meet. We saw:

- **Triangles** → Angles and edges inevitably yield Pythagoras.
- **Baseball** → Runs and outs force a scoreboard.
- **Quantum** → Measurement events plus amplitude data yield "collapse."

In each domain, we enumerated facts, letting **aggregator constraints** unify them. No special "theorem," `updateScore()`, or `collapseWavefunction()` calls were ever needed. This universal pattern underpins the **Conceptual Model Completeness Conjecture (CMCC)**:

> "Any domain—geometric, athletic, quantum, or otherwise—can be fully expressed by enumerating five primitives in a single snapshot-consistent model: **Schema (S), Data (D), Lookups (L), Aggregations (A), and Lambda-Calculated Fields (F).** Contradictions never commit, so you get a consistent system, automatically."

Why This Matters

- **Eliminates Stepwise Confusion**: You don't imperatively update geometry or baseball scores; you just record new facts (edge lengths, run events, measurement outcomes), and the aggregator logic enforces consistency.
- **No Partial States**: You can't have "three outs, but the scoreboard shows the same half-inning still in progress." Contradictions are blocked outright.
- **Ties Domains Together**: Because we never wrote specialized code for geometry vs. baseball vs. quantum, you could integrate multiple domains under one consistent approach.

Bigger Possibilities

This same approach can handle advanced sabermetrics, multi-qubit entanglement, or large-scale concurrency—*all by enumerating facts and aggregator-based constraints.* If the domain forms contradictory statements, the system simply refuses to finalize them—like a “right triangle” that tries to break a2+b2=c2a^2 + b^2 = c^2a2+b2=c2. In a concurrent environment, you can record partial facts or parallel events without risking partial-state contradictions at commit time.

Ultimately, the CMCC says this approach isn’t just a neat trick; it’s a universal method for capturing domain logic without “procedural illusions.” Truth emerges from structure. By enumerating what’s *true*, you can’t help but end up with

- c2=a2+b2c^2 = a^2 + b^2c2=a2+b2 for a right triangle,
- a scoreboard at three outs, or
- a wavefunction collapse upon measurement.

If this all sounds like a flavor of constraint-logic programming or relational modeling, you’re not wrong—but the CMCC viewpoint aims to unify everything in **one snapshot model**, across domains as disparate as geometry, baseball, and quantum.

Footer: Explore More

GitHub Repository
See a wide range of examples—from double-slit interference patterns to multi-qubit setups—in a purely data-first approach:
https://github.com/eejai42/conceptual-model-completeness-conjecture-toe-meta-model

Research Papers

- The Conceptual Model Completeness Conjecture (CMCC)
- BRCC / CMCC Turing Completeness Proofs
- Quantum CMCC (Q-CMCC) for multi-qubit entanglement and measurement events

These references dive deeper into concurrency, partial knowledge, and how we unify seemingly disparate domains under a single aggregator-driven model. If you’re curious about the formal proofs or want to reproduce double-slit results in Python, the GitHub and Zenodo links have it all.

Thanks for reading! If you had that moment of realization—“maybe the Pythagorean theorem, a scoreboard, and quantum measurement are all the same *kind* of story”—then you’ve glimpsed how powerful enumerating facts can be.