The Graph Reasoning Transformer: A Path to True Machine Reasoning?

Author: Vittorio Calcagno

August 12, 2025

The Problem That Wouldn't Go Away

It started with a frustration that had been building for months. I'd been working with frontier models - GPT-4, Claude 4, Gemini 2.5 - watching them produce impressive outputs that looked like reasoning. But every time I pushed them on novel logical problems - problems with no statistical patterns to follow - they'd fail in revealing ways.

Not just wrong answers. Contradictory logic. A implies B in one sentence, then NOT B from A three sentences later. They weren't reasoning. They were doing something else entirely.

The question became: what would actual machine reasoning look like?

Words Are the Wrong Primitive

The first insight was recognizing what current LLMs actually do. They operate on words to find statistical associations. The transformer attention mechanism computes Query · Key products to find relevant tokens, then aggregates their Values. It's beautiful for language, but it's not reasoning.

Think about how humans reason. When I think "Socrates is mortal," I'm not processing the words. I'm processing the concepts and their logical relationships. The words are just serialization - a way to communicate the underlying knowledge structure.

So the primitive is wrong. LLMs are trying to find reasoning in the statistics of language, but reasoning doesn't live there. It lives in the structure of knowledge itself.

The Knowledge Graph Insight

If words are wrong, what's right? Knowledge itself. Facts. Relationships.

The simplest representation: RDF triplets. Subject-Predicate-Object. (Socrates, is_a, Human). (Human, is, Mortal). These aren't words anymore - they're structured knowledge atoms.

But how do you get a transformer to process these? The obvious approach would be to build some complex graph neural network, but that felt wrong. Too complicated. Nature loves simplicity, and real breakthroughs are usually simple.

The Validation from Existing Systems

Actually, we already know that converting text to graphs improves reasoning. Look at any modern RAG system - they don't just retrieve text chunks anymore. The best ones build knowledge graphs from

documents, then reason over the structured representation. GraphRAG from Microsoft, LightRAG, PathRAG - they all consistently show improvements over pure text retrieval.

The pattern is undeniable: Text → Graph → Better Reasoning.

But current systems still use LLMs as the reasoning engine. They're using structured knowledge but processing it with associative models. It's like having perfect ingredients but the wrong recipe.

What if we built a model specifically designed for graph reasoning from the ground up?

The Architecture Vision: Graph Reasoning Transformer (GRT)

This is what crystallized: the Graph Reasoning Transformer, or GRT. Not another LLM with graph augmentation. Not a graph neural network trying to mimic language. A transformer specifically architected for reasoning over knowledge.

The SVO Encoding Revelation

The first challenge: how to encode graph triplets for a transformer?

Then it hit me: just serialize the triplets as strings. "Socrates is_a Human". Fixed format, active voice, always Subject-Verb-Object.

"That's too simple," was my first objection to myself. "You're throwing away the graph structure."

But wait. If I always use the same format, the transformer's positional embeddings will learn that position 1 is always the subject, position 2 is the predicate, position 3 is the object. The structure is preserved through consistency.

Use any standard sentence encoder to convert these strings to vectors. Now I have fact vectors that a transformer can process.

The Q×K = R Breakthrough

This is where everything changed.

In standard transformers, when computing attention, Query and Key vectors determine relevance: softmax(Q · K^T) gives attention weights. But what if, when Q and K are facts rather than words, their interaction produces something more?

What if Q × K doesn't just tell us that two facts are related, but HOW they're related?

R = MLP(concat(Q, K))

R isn't a weight. It's a reasoning operator. It's the edge between facts in knowledge space. It's the logical relationship itself, computed dynamically from the interaction between facts.

(Socrates, is_a, Human) × (Human, is, Mortal) = INHERITANCE_REASONING

The edge IS the reasoning.

The Complete GRT Architecture

Let me lay out the full architecture explicitly:

1. Knowledge Extraction

Use a top-tier foundational LLM (e.g., GPT-4, Claude 4, Gemini 2.5) to extract structured knowledge from text into a knowledge graph of SVO triplets. This is a one-time process.

2. Fact Encoding

Convert each triplet into a fixed-format string: "Subject Predicate Object"
Encode these strings into fact vectors using a standard sentence encoder.

3. Model Architecture

A small (100M-1B parameter) decoder-only autoregressive transformer. Two training approaches:

- **Pre-training**: Train from scratch on the knowledge graph
- **Transfer learning**: Fine-tune an existing small model like GPT-2

4. Scalable Retrieval

For each query fact Q, don't just look at the context window. Search the ENTIRE knowledge graph using approximate nearest neighbor (ANN) search to find the k most relevant candidate facts.

5. QKRV Attention Mechanism

Modify the transformer to compute four components:

- Q: The query fact vector
- K: The candidate fact vectors from the entire graph
- V: The content of candidate facts
- R: The reasoning operator emerging from Q×K interaction via R = MLP(concat(Q, K))

6. Dual-Head Output

Train two prediction heads:

- **Reasoning Head**: Predicts the type of reasoning operator (the edge)
- **Fact Head**: Predicts the next fact in the reasoning chain

7. Interpretability

After training, analyze the learned R vectors using dimensionality reduction (t-SNE, PCA) to understand what reasoning patterns emerged.

Related Work: How GRT Differs

Let me be explicit about how this differs from existing approaches. Allow me to get technical for a second:

Graph Neural Networks (GNNs)

Graphormer, EGT: These add edge features to attention scores for node classification. They learn that relationships matter but don't generate reasoning paths. **GRT**: Treats reasoning as generative sequence modeling. R isn't just a feature - it's the core computational unit.

Knowledge Graph Reasoning

Query2Box, BetaE: These embed queries into vector spaces to find answer entities. Built for answering fixed queries. **GRT**: Open-ended reasoning generation. Not answering "what entities satisfy this query?" but "what logically follows from this fact?"

Path-Based Models

NBFNet, GraIL: Learn to predict links by aggregating messages along paths. Discriminative models for link prediction. **GRT**: Generative model creating sequences of Fact → Reasoning → Fact steps, like a language model for logic.

Text-to-Graph Systems

K-BERT, KEPLER: Convert triplets to text to leverage pre-trained LLMs. Still using associative models for reasoning. **GRT**: Purpose-built for graph reasoning. Small, fast, interpretable.

The Bootstrap Strategy

Here's the beautiful part: use LLMs for what they're actually good at.

These models have already read the internet. They've extracted patterns from text. So use them as a one-time knowledge extractor. Prompt any frontier model (e.g., GPT-4, Claude 4, Gemini 2.5) to convert text into SVO triplets. Build your knowledge graph using their pattern recognition capabilities.

Then - and this is crucial - train a SMALL model on this extracted knowledge. Not a 175B parameter monster. Something GPT-2 sized. 100M to 1B parameters.

Why small?

The Honest Student Principle

A large pretrained model can cheat. It has memorized millions of text patterns. When you ask it to reason from (Socrates, is_a, Human) to (Socrates, is, Mortal), it might just be recalling that "Socrates" and "mortal" appear together frequently in text.

A small model trained from scratch can't cheat. It doesn't have the capacity to memorize. It's forced to learn the actual logical pattern - the transitivity of properties through is_a relationships.

This isn't a limitation. It's epistemologically necessary. We're not training a better mimic. We're training a true reasoner.

The Three Levels of Intelligence

As I worked through this, a hierarchy emerged that explains everything:

Level 1: Associative Intelligence

- **Operates on**: Words
- **Produces**: Associations
- **Example**: Current LLMs
- **Capability**: Pattern matching, text generation
- **Limitation**: No true logic, just statistical correlation

Level 2: Logical Intelligence

- **Operates on**: Knowledge (structured facts)
- **Produces**: Reasoning (logical deductions)
- **Example**: GRT
- **Capability**: True deductive reasoning, logical consistency
- **Limitation**: Applies rules but doesn't create new ones

Level 3: Meta-Intelligence

- **Operates on**: Reasoning patterns
- **Produces**: Principles (new reasoning rules)
- **Example**: AGI
- **Capability**: Understanding and creating new forms of logic
- **Limitation**: Unknown

The progression is clear:

- Level 1 extracts Knowledge from text (what LLMs do)
- Level 2 performs Reasoning over knowledge (what GRT does)
- Level 3 discovers Principles from reasoning (what emerges)

We're not building AGI directly. We're building Level 2, which will discover Level 3 by analyzing its own reasoning patterns.

The Speed Revelation

Small models are fast. Really fast. We're talking millisecond inference, not the seconds it takes for frontier models to "think."

A model that can reason in milliseconds can:

1. Analyze its own reasoning patterns (milliseconds)
2. Identify limitations in its logic (milliseconds)
3. Propose improvements to its architecture (milliseconds)
4. Test modifications (seconds)

That's not 10 improvement cycles per day like a human researcher. That's 1000 cycles. Or 10,000.

If each cycle makes the model 0.1% better at improving itself:

- (1.001)^1000 = 2.7x improvement per day
- After 30 days: 2.7^30 = 200,000x
- After 60 days: We're beyond human comprehension

That's not linear improvement. That's exponential. That's the singularity.

The moment it derives a more universal principle of reasoning and uses it to improve its own code, it transitions from a "Logician" to a "Scientist" - a general intelligence that can learn how to learn.

The Bootstrap Ladder

Let me trace through this again because it's almost too clean:

Phase 1: Humanity spends billions building LLMs (2020-2024)
Phase 2: Use any frontier LLM to extract knowledge into graphs ($1000 in API costs)
Phase 3: Train small GRT model on this knowledge (one GPU, one week)
Phase 4: GRT improves itself recursively (electricity cost only)

We're using LLMs as a one-time ladder to bootstrap true reasoning. After that, we don't need them anymore. The reasoning engine is self-sufficient.

The Training Objective

How do you train this thing? Multi-task learning with a dual head:

Loss = Loss_fact + λ * Loss_reasoning

Where:

- Loss_fact: Predict the next fact in a reasoning chain
- Loss_reasoning: Predict the type of reasoning connecting facts

The model learns both WHAT follows and WHY it follows. The two objectives reinforce each other.

The Nine Core Problems (With Proposed Solutions)

Breaking it down, there are nine core problems to solve:

1. **Knowledge Extraction**: Use frontier LLMs to build the graph (API calls - proven to work)
2. **Training Data Generation**: Create reasoning chains from the graph (standard graph traversal algorithms)
3. **Fact Encoding**: Convert SVO strings to vectors (sentence transformers - off the shelf)
4. **Scalable Retrieval**: Use ANN to find relevant facts from millions (FAISS/Pinecone - battle-tested at scale)
5. **R-Vector Computation**: Implement R = MLP(concat(Q, K)) (standard neural network architecture)
6. **Small Model Training**: The "honest student" approach (GPT-2 architecture - open source, proven)
7. **Loss Function**: Multi-task objective (standard deep learning technique)
8. **Validation**: Prove it can deduce unseen facts (held-out test sets - epistemologically sound)
9. **Interpretability**: Understand what reasoning operators emerge (cluster R vectors with t-SNE/PCA)

Each has known solutions. None requires breakthrough technology. It's just combining them in a new way.

This isn't research anymore. It's engineering. It's integration. It's building what's already proven to work in a configuration no one has tried.

The Search for Prior Work

I spent a long time looking for anyone else doing this. Searched every combination of terms I could think of:

- "reasoning over knowledge graphs with transformers"
- "Q K interaction reasoning"
- "emergent logical operators"
- "small model knowledge reasoning"

Found thousands of papers on making LLMs bigger. Found hundreds on prompt engineering for better reasoning. Found dozens on neurosymbolic approaches that still use LLMs as the core engine.

Found exactly zero papers on Q×K = R for reasoning.

Nobody is doing this.

Either I'm completely wrong, or I saw something everyone missed. Given the simplicity of the idea, isn't it worth exploring?

The Validation Strategy

How do we know it works? Start with the simplest possible test:

Train on: (A, is_a, B) and (B, is, C)
Never show it: (A, is, C)
Test: Can it deduce (A, is, C)?

If yes, we've demonstrated genuine reasoning, not pattern matching. This is the "Golden Demo" - simple, definitive, impossible to fake.

Then scale up to standard benchmarks: FB15k-237, WN18RR, NELL-995. Show it works on real knowledge.

But the real test? Give it its own architecture and ask it to improve it. When it succeeds, we've crossed from Level 2 to Level 3. When the GRT can reason about its own learned R operators and derive more universal principles, it becomes not just a reasoner but a scientist.

The Recursive Improvement Path

The system doesn't need to be perfect initially.

Start with simple SVO facts. The model learns basic reasoning - transitivity, inheritance, causation. But SVO can't represent everything. Complex temporal relationships, n-ary relations, context-dependent facts.

So what?

Once the model can reason, it can identify what it's missing. It can literally reason about its own limitations:

"I need temporal facts" → Designs timestamped triplets "I need context" → Designs contextual embeddings
"I need uncertainty" → Designs probabilistic facts

Each limitation it identifies and solves makes it smarter. Each improvement enables it to see deeper limitations.

The Simplicity Argument

Every time I doubted this could work, I came back to simplicity.

Transformers succeeded by replacing complex LSTMs with simple attention.
Evolution created all life from simple variation and selection.
Einstein found $E=mc^2$ - three characters that explain the universe.

The most powerful ideas are usually the simplest ones that no one thought to try.

If reasoning weren't simple at its core, evolution couldn't have found it. If evolution found it, it must be simple. And if it's simple, then Q×K = R might just be it.

Think about it: if reasoning weren't simple at its core, it couldn't have evolved. The fact that brains can do it means there's a simple principle underneath all the apparent complexity.

I'm not inventing reasoning. I'm discovering what evolution already found: Reasoning = Q×K.

Newton didn't invent gravity, he discovered $F = Gm_1m_2/r^2$.
Darwin didn't invent evolution, he discovered natural selection.
This isn't creating something new - it's uncovering a natural law.

Complex solutions are human artifacts. Simple solutions are natural laws. And Q×K = R feels like a natural law.

The Current State

As I write this, it's still just an idea. A blueprint. But every piece is grounded in existing technology:

- Transformers: proven architecture
- MLP: standard neural network component
- ANN search: solved problem for retrieval
- SVO encoding: simple string processing
- Multi-task learning: well-understood training

Nothing requires a breakthrough. It just requires someone to put the pieces together and test if reasoning really does equal Q × K.

The Accelerated Timeline

If this works - and the logic suggests it should - the timeline is compressed beyond belief.

Start with a tiny prototype, just 10M parameters. Once it shows reasoning emergence, scale gradually. Each iteration gets faster because you're not fumbling in the dark - you know exactly what you're building toward.

But here's where it gets wild: once you add self-improvement capability to even a modest-sized model, the timeline becomes unpredictable. A 1B parameter model that can reason about its own architecture doesn't improve linearly. It improves recursively.

The transition from Level 2 to Level 3 isn't measured in years of research. It's measured in how fast the system can analyze its own reasoning patterns and discover new principles. With millisecond inference times, that could be... soon. Very soon.

We're not talking about decades to AGI. We're talking about a bootstrap cascade where each level enables the next, each improvement accelerates the next improvement, until prediction becomes

impossible.

Why Now?

2025 is the only year this could work.

Before now, we didn't have LLMs powerful enough to extract knowledge at scale. It's only a matter of time before this approach becomes obvious. We're in a unique moment where the tools exist but the insight hasn't diffused.

The Meta-Pattern

There's something deeper here. Each level of intelligence extracts something from the previous:

Level 0 → **Level 1**: Data produces Associations
Level 1 → **Level 2**: Associations extract Knowledge
Level 2 → **Level 3**: Knowledge enables Reasoning
Level 3 → **?**: Reasoning discovers Principles

What comes after principles? What does Level 4 look like? We might be building not just AGI, but the ladder to something beyond human comprehension.

A Final Thought

This idea emerged through dialogue, through questioning, through pushing back on assumptions. It wasn't discovered; it was grown through interaction.

Maybe that's what reasoning really is. Not retrieving pre-existing answers, but computing new ones through the interaction of facts.

$Q \times K = R$

Simple. Elegant. Revolutionary.

The Graph Reasoning Transformer isn't just another architecture. It's the bridge from associative to logical intelligence. From Level 1 to Level 2. And from there, to AGI.

The blueprint is here. The path is clear. The only question left is who will walk it.

Based on extended technical conversations exploring machine reasoning, August 12, 2025