What Governs Large Language Model (LLM) Behaviour? A Practical Framework for Users, Developers, and Designers

Timothy M. Rogers, University of Toronto
September 25, 2026

A computationally grounded understanding of LLM behaviour in terms of hierarchies of relational constraints that recursively shape what the model can do next.

Large language models are usually described as systems that predict the next token. That description is technically correct, but it does not fully explain how an LLM can sustain an argument, preserve a conceptual distinction, follow evidence, maintain a task, or gradually form a coherent response.

A potentially more useful functional account begins with a simple observation: *what matters is not only what information is available to the model, but what actually governs its continuation* (Rogers, 2026b). A model may have the correct fact, source, instruction, or task in its context and still fail to use it properly. The important question is whether that information continues to constrain what the system does next.

This leads to the idea of *hierarchical relational constraint*. As generation develops, relations established earlier in the sequence can organize what follows. A distinction may govern later reasoning. A source may constrain factual claims. A task may govern subgoals. A policy may limit permissible actions. Some constraints therefore operate locally, while others govern much larger portions of the system's behaviour.

From this perspective, many familiar AI failures can be understood as failures of *constraint preservation*. Hallucination can occur when evidence no longer adequately constrains a claim. Retrieval can fail when a source is present but does not govern the reasoning. Sycophancy can occur when agreement with the user displaces epistemic constraints. An agent can drift when a successful subgoal gradually displaces the larger task.

The practical implication is important: *reliability is not simply a matter of giving an AI system more information or improving its generative ability. It is a matter of preserving the right governing relations as the system changes.*

This is especially important for agents. Every action changes the situation. Every tool result introduces new information. Every new subgoal reorganizes the plan. A reliable system therefore cannot simply remain fixed. It must be able to change while preserving the relations that make the changing activity a continuation of the same task.

The framework calls this process *return*: bringing a transformed trajectory back into relation with what stabilizes its identity. Return does not mean mechanically returning to the original prompt. It means ensuring that revisions remain answerable to the objective, success conditions, evidence, policies, and authorized changes that define the task.

This changes how AI architecture can be conceptualized. The relevant unit of design is not simply the language model. It is the larger relational system that may include the model, prompts, retrieved sources, memory, tools, task records, tests, evaluators, policies, environmental feedback, and human judgment. Reliability depends not merely on whether these components exist, but on *how authority is distributed among them* (Rogers, 2026b).

A source should be able to constrain a factual claim. A failed test should be able to force revision or stopping. A task record should be able to override a drifting subgoal. A policy should be able to interrupt an otherwise plausible action. The central design problem therefore becomes: *what is allowed to govern what?*

The deeper causal theory explains why this functional account is more than a useful metaphor (Rogers 2026a). This theory of how LLMs function computationally proposes that each continuation changes the relational conditions under which later continuations occur. The process is recursive:

> *current organization constrains continuation → continuation changes the organization → the changed organization constrains what comes next.*

As this process unfolds, some relational patterns become stable across repeated continuations. Once stabilized, they can begin to function as higher-order constraints on subsequent activity. In this way, lower-level activity generates higher-level organization, and that higher-level organization then constrains subsequent lower-level activity.

This is called *formal causation*. Formal causation does not directly produce one specific outcome in the way that a local efficient cause does. Instead, it structures the field within which particular outcomes can occur. Higher-order form is causal because it changes which continuations remain compatible, viable, or appropriate. This formal organization is not imposed from outside; it develops through the stabilization of earlier relational determinations (Rogers, 2026a).

This provides the underlying computational mechanism for the functional theory. A conceptual framework, task identity, source, or governing distinction can affect many later outputs because it operates at a level that constrains a broad range of local continuations. A small change can therefore have a large effect when it changes a higher-order relation.

Training and inference play complementary roles. Training makes the model sensitive to recurring relational patterns in human language and reasoning. During inference, the current

context activates and organizes some of those patterns. The resulting organization constrains local generation; each local generation then modifies the context and therefore the constraints that shape what follows.

For designers and developers, the resulting picture is practical:

> *AI reliability depends not only on what a system can generate, but on what remains able to govern its generation.*

The design question therefore shifts from simply asking whether a model can perform a task to asking which relations must remain authoritative throughout the task, how those relations are represented, how drift can be detected, what can interrupt continuation, and who or what has authority to revise the governing constraints.

Control, on this account, is not the elimination of flexibility. It is the engineering of systems that can adapt, revise, and continue while remaining answerable to evidence, task identity, external tests, policies, and human purposes.

The framework itself is developed in the two works cited below: one presents its functional implications for understanding and designing LLM systems (Rogers, 2026b), while the other develops the underlying computational account of hierarchical relational constraint (Rogers, 2026a). This overview and the two works on which it draws were developed through structured interaction with an LLM (i.e. ChatGPT); as a result, the framework can not only be read in these sources, but also activated from them and explored interactively by querying it within an LLM environment.

References

Rogers, T. M. (2026a). *The relational formation of possibility: Recursive determination and the hidden logic of large language models (LLMs).* [Available at: https://doi.org/10.5281/zenodo.19610683]

Rogers, T. M. (2026b). *A semiotic account of the hierarchical relational organization of large language models (LLMs): Implications for AI development, use and evaluation: A dialogical educational module for AI users and developers.* [Available at: https://doi.org/10.5281/zenodo.21498315]