Likologia

From: **Joe Maxwell from Joe Maxwell**
joesystemica@substack.com

To: **joseph.maxwell02@proton.me**
joseph.maxwell02@proton.me

On: **Tuesday, 26 May 2026 at 3:02:35**

Likologia

A formal grammar for systems defined by constraint

JOE MAXWELL
MAY 26

READ IN APP

Systems do not usually fail suddenly.

They fail when recovery can no longer keep up with drift.

That is the simplest way into Likologia.

Most of the time, when we talk about systems, we start with what they are made of. We name the parts. We describe the mechanism. We explain how the components interact. That works when the system is stable, because stable systems make component-level explanation look enough.

But under pressure, that changes.

The parts may stay the same while the behaviour shifts.

A material under load can look fine until it yields.
A person under sustained stress can appear functional until they shut down.
A hospital can keep operating while backlog grows invisibly.
An AI model can sound coherent while drifting away from the question it was meant to answer.

In each case, the visible collapse looks sudden.

Usually it is not.

What appears suddenly is the boundary crossing.

The drift was already happening.

Likologia is my name for the grammar underneath that pattern.

Not a theory of everything. Not a replacement for physics, medicine, psychology, engineering, computing, or politics. Not a claim that all systems are secretly the same thing.

A grammar.

A way of describing systems as systems before the language of a particular domain takes over.

The canonical version defines Likologia as the study of systems independent of domain: not physics specifically, not cognition specifically, not computation specifically, not society specifically, but how systems exist, stabilise, fail, transition and recover under constraints.

That last word matters.

Constraint.

A system is not only what it contains.

It is what it is allowed to do.

The first move

The normal question is:

what is this system made of?

Likologia asks:

what states can this system occupy, what transitions are allowed, and what constraints limit those transitions?

That is the base grammar.

A system has possible states.

It can move between some of them.

It cannot move freely.

That is enough to begin.

The formal draft reduces this to:

System = (States, Transitions, Constraints)

States are possible configurations. Transitions are allowed changes. Constraints are the limits on those changes.

This sounds simple because it is meant to be simple.

If something has distinguishable states, allowed transitions and constraints on those transitions, it can be treated as a system in this grammar. If it does not, then either it is not a system in this sense, or the model has not found the system yet.

The point is not to erase the domain.

A furnace is not a mind.
A mind is not an economy.
An economy is not an AI model.

But each can stabilise.
Each can drift.
Each can collapse.
Each can recover.

The mechanisms differ.

The behavioural grammar can still match.

That is the opening.

Systems move

A system is not just a collection of parts.

It is a moving position inside a space of possibilities.

At one moment, the system is here. Later, it is somewhere else. The path between those positions is its trajectory.

That word matters because collapse is not only an event. It is often the

end of a trajectory that was already moving toward a boundary.

In the formal version, this appears as a simple evolution rule:

$$x(t + 1) = f(x(t), C, P)$$

where x is the current system state, C is the active constraint set, and P is accumulated pressure or load.

That is not there to make the piece look mathematical.

It says something very concrete:

The next state of a system depends on where it is now, what constraints are active, and what pressure has accumulated.

That is the difference between a static description and a dynamical one.

A static description asks:

what does the system look like?

A dynamical description asks:

where is the system moving, and what is changing its allowed movement?

That shift is small, but it changes everything.

The shoebox

Imagine something complicated inside a shoebox.

You cannot open the box.

You can only look through a small hole.

You can shine light in from one angle. You can poke it gently with a stick. You can rotate the box slightly, but not enough to see everything. You have limited time. The lighting is imperfect. The object inside may be simple or extremely complex, but you never get the object directly.

You get the object through constraint.

You are not observing the system.

You are observing the system under restricted access, distorted signal,

limited interaction and time pressure.

This is not just an image.

It is a warning.

Because if the output looks messy, you might blame the object. But the mess may be coming from the viewing condition.

If a person cannot explain themselves clearly under pressure, that does not prove the internal model is absent.

If a material behaves differently under one test setup, that does not mean the material has changed identity.

If an AI model gives a worse answer after being loaded with contradictory context, that does not mean the model suddenly became stupid.

Output is not structure.

Output is structure under constraint.

The longer Likologia draft uses this as the core reverse-inference problem: systems are often inferred from outputs, but outputs already reflect access limits, noise, interaction limits and time constraints.

That is why Likologia starts with constraint.

Not because components do not matter.

Because components are often not what you are actually seeing.

Stability is not stillness

A stable system is not one that never changes.

A stable system is one that can absorb disturbance without losing its regime.

A bridge flexes.
A body adapts.
A mind shifts attention.
A queue absorbs a spike.
A model handles ambiguity.

Small disturbances enter. The system bends, redistributes, dissipates,

regulates, updates, or repairs. It remains inside a region where return is possible.

That region is a basin.

Not a literal hole. A behavioural region.

Inside the basin, recovery works.

Outside it, the old restoring forces are no longer enough.

This is why stability is local.

A system can be stable under one set of conditions and unstable under another. “It worked before” is not proof that it will work now, because the system may no longer be in the same region.

This is basic systems dynamics.

The system state moves through a constrained space. Some regions pull it back toward stability. Some regions amplify deviation. Some boundaries separate one operating regime from another.

Cross the wrong boundary and the same inputs no longer produce the same outputs.

The system has not merely changed amount.

It has changed regime.

Drift

Drift is what happens when unresolved perturbation starts accumulating.

At first, nothing dramatic needs to happen.

The system still works.

The material still holds.
The person still replies.
The hospital still opens.
The model still sounds fluent.

But recovery is taking longer. Variance is rising. Small disturbances linger. Sensitivity increases. The system becomes more dependent on favourable conditions.

This is the dangerous region because it is easy to misread.

From the outside, outputs may still look acceptable.

Internally, margin is disappearing.

The experimental protocol outline makes this measurable: across thermal, cognitive, queue and network systems, Likologia predicts pre-collapse signals such as increased recovery time, increased variance, increased sensitivity and hysteresis before collapse.

That is the key.

Collapse should not be treated as the first sign.

By the time collapse is visible, drift has usually been speaking for a while.

We just were not measuring the right thing.

The governing condition

The central condition is simple:

drift_rate < recovery_rate

When recovery is faster than drift, the system remains stable.

Disturbances arrive, but they are resolved. The system can absorb, reset, redistribute, repair or compensate before the next load overwhelms it.

As pressure increases, the relationship tightens.

drift_rate approaches recovery_rate

This is the drift regime.

The system still functions, but recovery no longer comfortably outruns disturbance. Perturbations begin to persist. The system becomes more sensitive to additional load.

Then the relationship reverses.

drift_rate ≥ recovery_capacity

At that point, the prior regime cannot be maintained.

This is why systems do not fail simply because pressure exists.

Pressure is normal.

Systems fail when recovery cannot keep up with pressure-driven drift.

The Likologia outline treats this as the core spine: stability depends on drift rate remaining below recovery rate, and collapse occurs when drift meets or exceeds recovery capacity.

That is the whole thing compressed.

A system remains stable when recovery outpaces drift.

It collapses when interacting pressures push it beyond recoverable regimes.

Why collapse feels sudden

Collapse often feels sudden because boundary crossings are visible and drift is not.

You notice the fracture, not every microstructural defect that made the fracture likely.

You notice the shutdown, not every recovery window that was missed.

You notice the queue collapse, not every small backlog that stopped clearing.

You notice the hallucination, not every tiny reasoning drift that made the final answer unstable.

In dynamical terms, the system moves continuously until it crosses into a new regime. The transition may look discontinuous from the outside because the output changes sharply.

But the movement toward that transition was often gradual.

That is why static snapshots mislead.

They show where the system is.

They do not always show how close it is to the boundary.

Collapse is not one thing

People say “collapse” as if it names a single event.

It does not.

Systems fail in different shapes, and the shape matters because the recovery path depends on it.

A **runaway collapse** happens when feedback amplifies itself. The more the process happens, the faster it happens. Thermal runaway is the obvious engineering example. Panic spirals can behave similarly. So can narrative drift in an LLM when each invented claim becomes context for the next invented claim.

A **saturation collapse** happens when capacity is exceeded. The system does not explode. It fills. Then it cannot process more. Queues grow. Energy runs out. No movement is possible because the system is at its limit.

A **fragmentation collapse** happens when coherence is lost. Parts may still function, but they no longer coordinate. A cracked material, a broken team, a disorganised cognitive state and a corrupted information system can all fail this way.

An **oscillatory collapse** happens when the system cannot settle. It overcorrects, swings back, overcorrects again and never returns to stable regulation.

These are not just labels.

They tell you what kind of intervention makes sense.

Runaway needs feedback suppression.
Saturation needs load reduction or capacity increase.
Fragmentation needs coherence restoration.
Oscillation needs feedback stabilisation.

The formal Likologia draft makes this point directly: systems do not simply fail, and identifying the dominant collapse mode is critical because different collapse structures imply different recovery constraints.

That is why “fix the system” is usually too vague.

Fix what?

Feedback? Capacity? Coherence? Oscillation?

The answer changes the intervention.

Recovery is not reversal

Recovery is often described as going back.

That is usually wrong.

A system that has crossed a boundary may not be able to retrace the path it took into collapse. The constraints have changed. Capacity has changed. Damage may have occurred. Resources may be depleted. The old basin may no longer be accessible.

Recovery means re-entry into a stable region.

Sometimes that is the old region.

Sometimes it is a new one.

A person recovering from burnout may not return to the old operating pattern. A service system after crisis may need restructuring, not just more effort. A material after damage may require repair, annealing or replacement. An AI conversation after drift may need re-anchoring, not "continue from here."

Recovery has costs.

It takes energy.
It takes time.
It requires available paths.
It depends on collapse type.

This is why forced recovery often fails. It tries to push output before the system has re-entered a region where output is recoverable.

In Likologia, recovery is not moral.

It is dynamical.

Can the system get back into a region where disturbance can be absorbed?

That is the question.

Pressure is rarely single

Another reason systems are misread is that pressure is rarely one clean

force.

Pressure can be additive, where loads simply stack.

It can be multiplicative, where one pressure makes another worse.

It can be threshold-based, where nothing much happens until a limit is crossed.

It can be oscillatory, where the timing of pressure matters as much as its size.

It can be nonlinear, where small inputs suddenly produce large outputs because the system is already near a boundary.

That is why single-cause explanations often fail.

A system may not collapse because one thing was too much.

It may collapse because several things interacted in the wrong regime.

A body under poor sleep, infection, pain, social stress and autonomic instability is not dealing with five separate problems. It is dealing with a coupled load state.

A technical project under deadline pressure, unclear requirements, fragile modelling assumptions and poor documentation is not dealing with four separate risks. It is dealing with interaction dominance.

A machine-learning model with noisy data, target leakage, unstable descriptors and overconfident interpretation is not simply "a bit uncertain." It may be in a regime where explanation is no longer earned.

This is why systems dynamics matters.

It tells us to look for interaction, accumulation, feedback, thresholds and recovery rates.

Not just parts.

Not just causes.

Motion under constraint.

Why this matters

Likologia is useful because it stops us making the same mistake in

different languages.

In engineering, the mistake is trusting a design because it works under one condition.

In medicine, the mistake is treating symptoms as isolated components instead of system states.

In cognition, the mistake is judging output without checking access, noise, stakes and recovery.

In AI, the mistake is treating fluent language as stable reasoning.

In institutions, the mistake is assuming a system is healthy because it has not visibly failed yet.

These are different domains.

But the warning is similar:

do not mistake apparent function for stability.

A system can function while drifting.

That sentence explains a lot.

It explains why collapse feels sudden.

It explains why people are surprised by failure.

It explains why snapshots mislead.

It explains why health, infrastructure, education, engineering and AI all need time-sensitive evidence, not just static outputs.

The experimental outline turns this into a testable claim: if Likologia is valid, precursor signals such as rising recovery time, variance, sensitivity and hysteresis should appear before collapse across multiple domains under controlled conditions.

That is the claim ceiling.

Not “this explains everything.”

More precise:

if this grammar is useful, it should help identify drift before collapse

becomes visible.

What would break it

A framework is only useful if it can fail.

Likologia weakens if systems can be found where behaviour is independent of constraint, where perturbations neither accumulate nor dissipate, where no regimes or boundaries can be identified, or where collapse occurs without measurable precursor under proper observation.

The formal draft states this directly: the framework must be revised or restricted if systems exist where behaviour is independent of constraint, perturbation does not accumulate or dissipate, or no identifiable regimes and boundaries exist.

That matters.

The aim is not to make a language so broad that nothing can disprove it.

The aim is to make a grammar sharp enough that its failures are visible.

The practical use

Once you see the pattern, the questions become simple.

Where is the system stable?

What is drifting?

What pressure is accumulating?

What recovery capacity exists?

What boundary is being approached?

What collapse mode is most likely?

What would recovery require?

What are we not seeing because our access is constrained?

That is already a better conversation than most systems get.

It does not replace expertise.

It disciplines it.

A thermal engineer still needs thermal knowledge.
A clinician still needs medicine.
A modeller still needs maths.
A policymaker still needs data.
An AI researcher still needs evaluation.

Likologia does not remove domain knowledge.

It gives domain knowledge a shared behavioural grammar.

That is the point.

A system is not just what it contains.

It is the space of states it can occupy, the transitions it can make, and the constraints that decide what becomes possible under pressure.

Stable.
Drifting.
Collapsed.
Recovering.

Different worlds.

Same warning.

Recovery must keep up.