// WHITE PAPER · PUBLISHED 27 JULY 2026

Reasoning Without a Reasoner

Six semantic-web tools, one workstation, and a knowledge graph that has to survive being wrong

Jason Newell · MAX Research Collective · Report WPR-MAX-016 · Published 27 July 2026
Licensed under the Creative Commons Attribution 4.0 International License (CC BY 4.0), https://creativecommons.org/licenses/by/4.0/
Original: https://www.maxresearchcollective.com/articles/reasoning-without-a-reasoner.html
Cite as: Newell, J. (2026). Reasoning Without a Reasoner. MAX Research Collective, WPR-MAX-016. https://www.maxresearchcollective.com/articles/reasoning-without-a-reasoner.html

MAX runs on one workstation. A single RTX 3090 Ti carries the language model, the voice path, and a Forward-Forward learner that keeps training while the rest of the system answers questions. Forward-Forward replaces backpropagation with two forward passes, and it matters here because it runs constantly and it wants the card. I spent this month on one question: whether to add a Java runtime to that machine. A Java runtime would let a Description Logic reasoner classify MAX's knowledge graph. Description Logic is the formal family underneath the Web Ontology Language, and a reasoner is the engine that computes what your stated facts imply.

The answer came out of the data rather than a benchmark. Three of the six tools weighed here are excellent at jobs MAX doesn't have. A fourth already ships in his exporter.

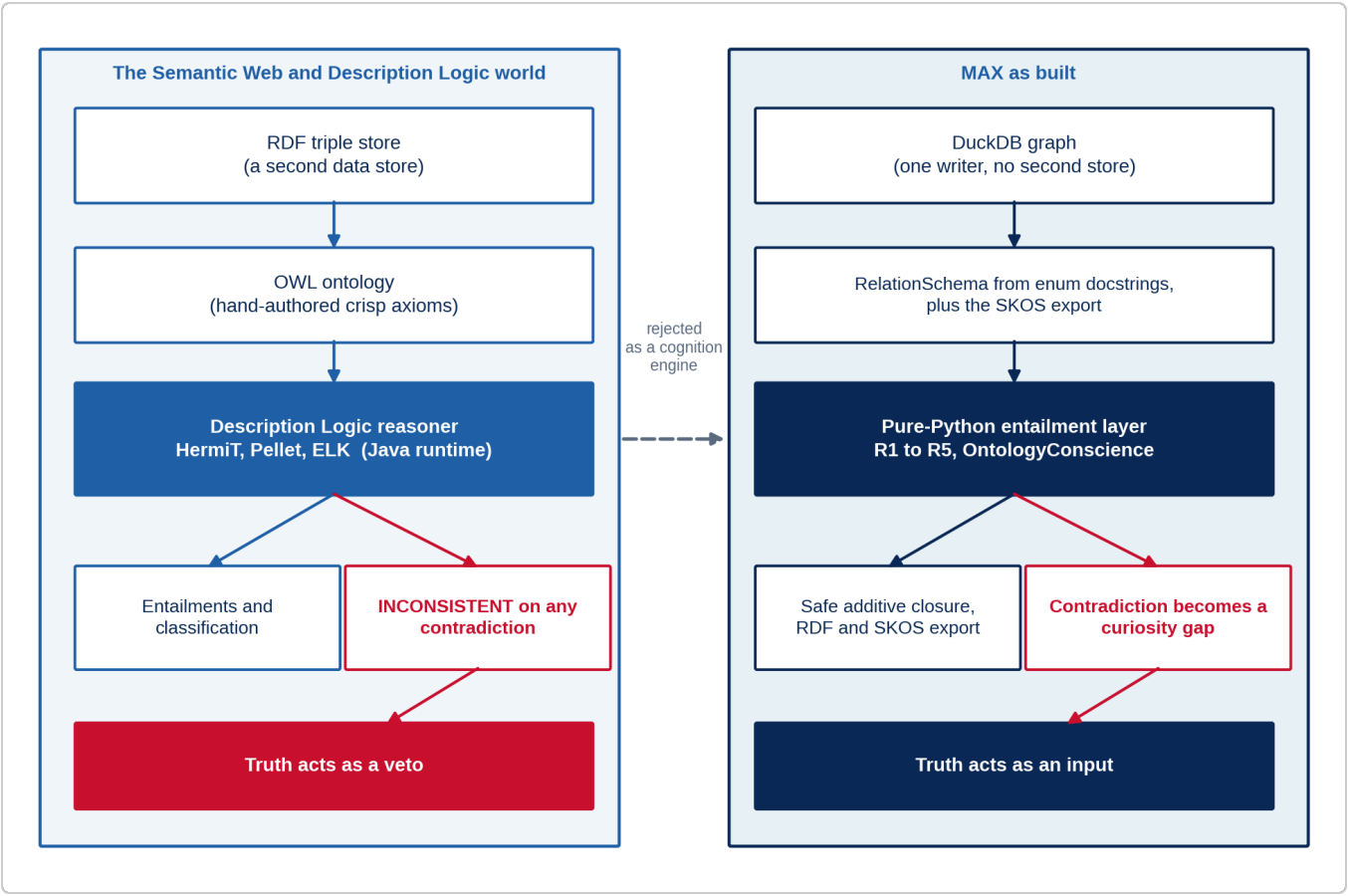


Figure 1. The dashed arrow across the middle is the decision; everything to the right of it runs without a second runtime.

Six tools, and only three of them are engines

RDF Schema, the Web Ontology Language, and the Simple Knowledge Organization System are languages. You write facts and axioms in them, and they can sit in storage forever with nothing computing over them. HermiT, Pellet, and ELK are engines. An engine reads an ontology and computes its entailments. An entailment is a fact that follows from the ones you stated. All three engines run on the Java Virtual Machine.

TOOL	KIND	WHAT IT GIVES YOU	SEMANTICS	RUNTIME
RDF Schema	Language	Classes, subclass relationships, property domains and ranges	Simple entailment, weak by design	None, it is data
Web Ontology Language	Language	Cardinality, disjointness, class expressions, property characteristics	The Description Logic SROIQ(D), open-world and monotonic	None, it is a language
SKOS	Language	Concepts, a broader and narrower relation, an associative relation, labels	Informal on purpose, since broader is not the subclass relation	None, it is data
HermiT	Engine	Hypertableau reasoning over the full dialect, from Oxford	Sound and complete for the full dialect	Java Virtual Machine
Pellet	Engine	Tableau reasoning, plus rules, plus justifications	Sound and complete for the full dialect	Java Virtual Machine
ELK	Engine	The tractable EL profile only, in polynomial time	Sound and complete for EL alone	Java Virtual Machine

That split decides the evaluation. A language costs storage. An engine costs a runtime, and on this machine a runtime is the expensive thing.

What each one is genuinely good at

RDF Schema gives you classes, subclass relationships, and property domains and ranges. It is decidable, it forward-chains with a handful of rules, and everything supports it. It also cannot express negation, cardinality, or disjointness. You cannot say that a person has exactly one birth mother, and you cannot say that two

classes share no members. Its entailment is weak enough that it misses most real modelling errors. None of that is a flaw when the job is small. Publish structured data on a web page and RDF Schema is the whole answer. Run a linked-data catalogue where subclass plus domain and range is all the structure you need, and no reasoner ever enters the loop.

The Web Ontology Language is expressive where RDF Schema is thin. Cardinality, disjointness, inverse properties, transitivity, class expressions, and equivalence are all available. Its main dialect rests on a Description Logic called SROIQ(D), and two of its properties surprise people arriving with application data. The first is the open-world assumption: a fact nobody stated is unknown, never false. The second is monotonicity, so a new fact never retracts a conclusion you already drew. Worst-case complexity is N2ExpTime-complete, about as bad as decidable gets, though real ontologies rarely reach the worst case. Getting value out of the language requires a reasoner, and every production-grade reasoner runs on the Java Virtual Machine.

That is the right tool when the ontology is crisp and curated, and a contradiction means somebody made a mistake. Clinical terminologies live there. So do the product-configuration validators that answer whether a combination of options is permitted at all. So do aerospace and medical-device ontologies, where an inconsistency is a defect and refusal is the correct response.

The Simple Knowledge Organization System is weak on purpose, and the weakness is the point. It gives you concepts, a broader and narrower relation, an associative relation, and labels. Its broader relation carries no logical entailment, since it is not the subclass relation and was never meant to be one. Nothing type-checks, nothing proves consistency, and nothing can explode. That fits a thesaurus, a set of subject headings, or a tag hierarchy. It also fits MAX's six disciplines, thirty-seven subjects, and the topic tree beneath them, where "history is broader than the Second World War" is a useful human relationship and a poor logical claim. A reasoner handed that sentence as an axiom would be right to reject it.

Then the three engines, which are the part of this world that costs something to run.

ENGINE	SWEET SPOT	WHAT IT BUYS	WHAT IT COSTS	REACH FOR IT WHEN
HermiT	The full dialect	Hypertableau is efficient on many real ontologies, and nothing in the language is off limits	A Java runtime, worst-case intractability, and memory exhaustion on a hard schema	You need full-dialect classification over a mid-size, expressive research ontology
Pellet	The full dialect, plus rules and explanations	Rules, datatype reasoning, incremental classification, and a justification for every entailment	A Java runtime, an original AGPL license, and maintenance that has moved to forks	You need the dialect plus rules plus explanations, and the license clears
ELK	The EL profile alone	Polynomial time, parallel execution, and SNOMED CT classified in seconds	A Java runtime, and no negation, disjunction, or universal restrictions	You hold a very large biomedical terminology written to fit inside EL

All three are sound and complete, so each finds every entailment and invents none. They separate on expressivity, on speed, and on license. HermiT is Oxford's hypertableau reasoner, covering the full dialect, efficient on many real ontologies and able to exhaust memory on a hard one. Pellet adds rules, datatype reasoning, incremental classification, and justifications. A justification answers why a particular entailment holds. Its original license is AGPL, a copyleft license that reaches through a network boundary. That makes it a hazard for anything that might ship as a product. ELK handles only the tractable EL profile, which drops negation, disjunction, and universal restrictions. In exchange, ELK classifies SNOMED CT's roughly 350,000 concepts in seconds. A full-dialect reasoner on the same terminology can run for hours or fail to finish.

The pick follows from the profile and the size, and from nothing else. Huge and inside EL means ELK. Expressive and mid-size means HermiT. Rules and explanations, if the license clears, means Pellet or its Openllet fork.

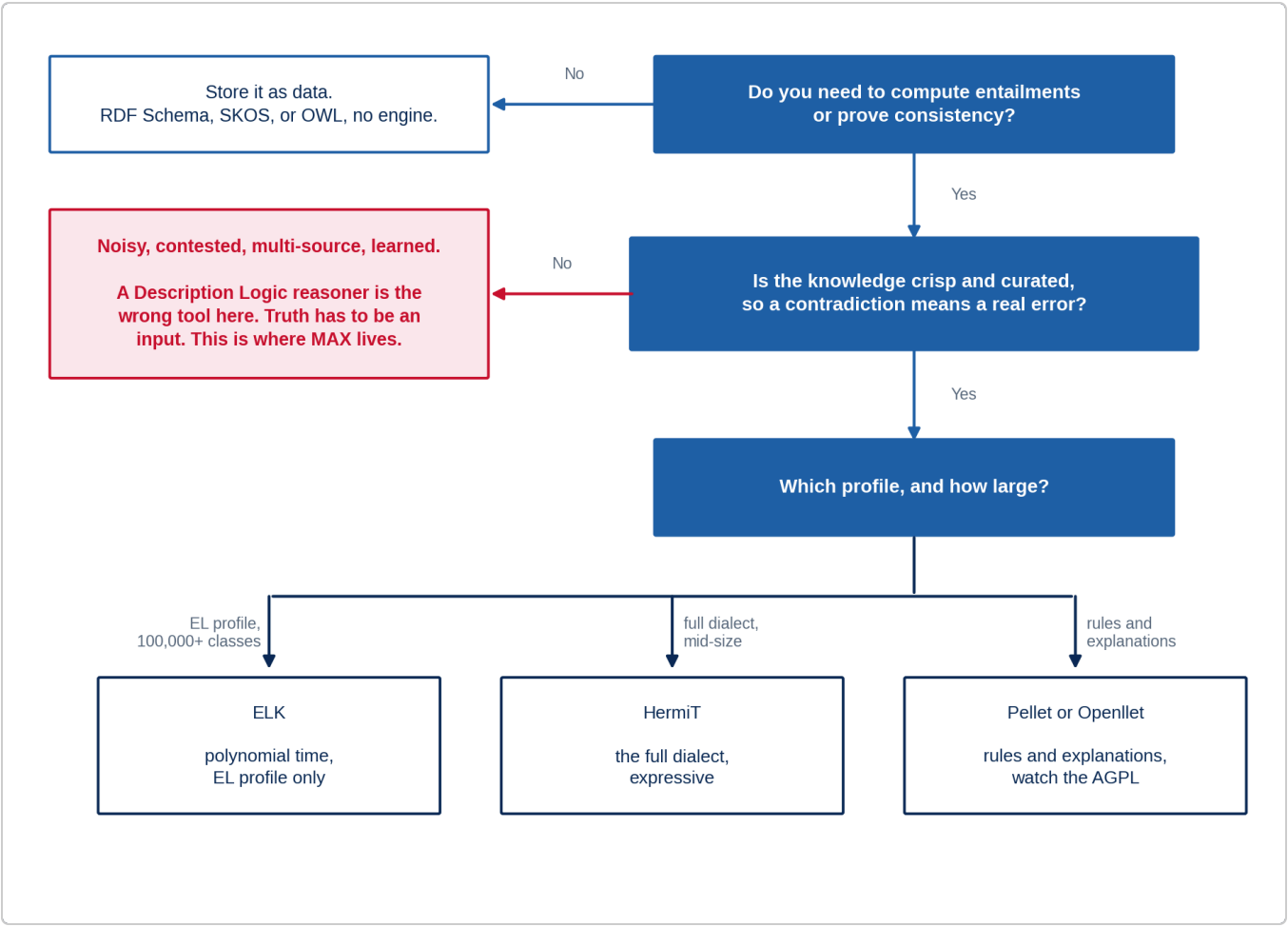


Figure 2. The pivot is the second question, since profile and size only start to matter once the knowledge is known to be crisp.

The disqualifier

Assert two contradictory facts to a Description Logic reasoner and it declares the dataset inconsistent. The finding lands on the whole dataset rather than on the two offending facts. From there the reasoner refuses to infer anything. On the classical reading it infers everything, because a contradiction entails every proposition. That behaviour is correct for a hand-built ontology, where a contradiction means a modeller erred and downstream work should stop until somebody hears about it.

MAX ingests 57 sources that disagree on purpose.

Perception is probability in his design rather than fact. Every claim carries a confidence number and a provenance string, and it sits on an epistemic ladder that records how it was learned and how far it has been promoted. Memory is a hint rather than a record, which is Rule 5 of the build. A forgetting curve decays stale claims and deletes none of them. Beliefs carry a stance, and contested is one of the stances on offer.

WHAT A DESCRIPTION LOGIC REASONER ASSUMES	WHAT MAX IS BUILT ON
A contradiction halts reasoning	Truth is an input rather than a veto, and perception is probability
A fact is either true or false	Every claim carries confidence, provenance, and a rung on the epistemic ladder
Axioms are permanent	A forgetting curve decays stale claims, and a belief carries a stance
One world, and it is consistent	57 sources that conflict on purpose, each one belief-stanced

Run a reasoner across that corpus and one of two things happens. It declares the whole thing inconsistent and does nothing useful, or the build firewalls a small clean subset for it to work over, which is a demo rather than a mind. So the most valuable output a reasoner produces, the finding that two facts contradict, arrives in the one form MAX has to handle backwards.

Backwards looks like this. A detected contradiction files a curiosity gap under a tension label. A conflict record penalises the confidence of whichever edge has weaker provenance, and it leaves the stronger edge alone. Tensions in medicine or law get framed for the person asking. Nothing is deleted and nothing is vetoed. For the reasoner, inconsistency means error. Here it means the next thing worth understanding. That inversion is only sane because memory was a hint to begin with.

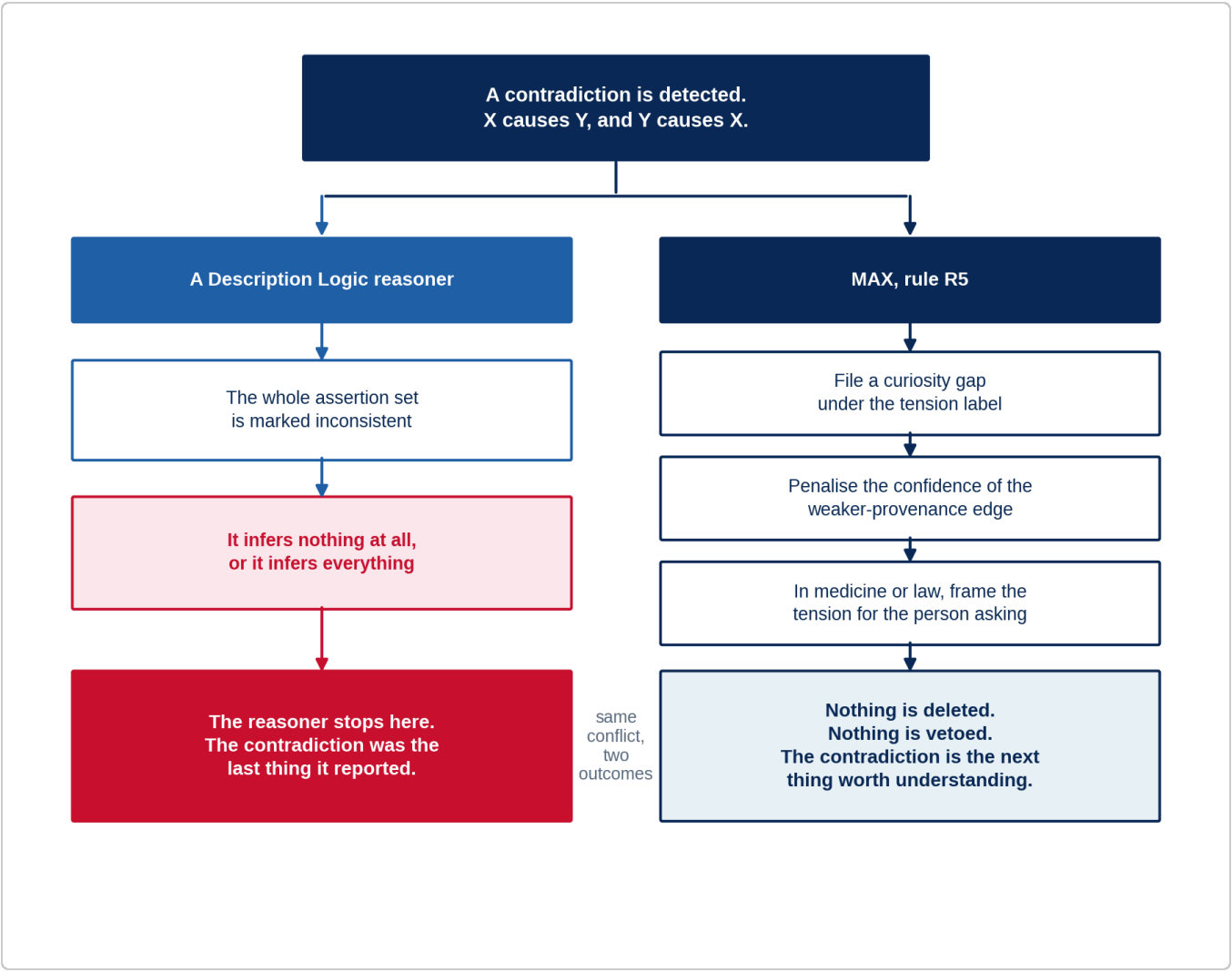


Figure 3. Both paths detect the same conflict, and only one of them is still working afterwards.

The rest of the arithmetic

Java is not banned in this house. Ollama already runs out of process and is not Python, because Rule 1 says a single core process rather than a single language. A reasoner running as a local health-checked service is the same architectural shape as Ollama. The door was open. Every other axis was still bad.

THE CONSTRAINT	WHY A DESCRIPTION LOGIC REASONER CONFLICTS WITH IT
Rule 1, local-first, a single core process	All three engines are Java. A second runtime brings its own garbage collector and heap pressure, competing with the language model, the learner, and the voice path.
One DuckDB writer	A reasoner works over an in-memory RDF store rather than DuckDB, so the build would need an export, a reasoning pass, a write-back, and a second store kept in sync. That solves the smaller half of the problem and leaves the sync unsolved.
Rule 2 and the commercial gate	Pellet's original license is AGPL, which is a hazard for a clinical product. Only Openllet and ELK are clean.
The stdlib-first direction	A Java dependency brings install fragility to a build that has been shedding dependencies rather than collecting them.
Operational simplicity	The supervisor, the watchdog, the self-repair manager, and the host-hygiene stack would all have to learn a Java process.
Learned rather than declared	MAX learns which relations are inverses by watching them turn up as pairs. A reasoner wants those declared by hand, which runs against the substrate underneath him.

The recorded decision is no Java runtime for this arc. If a future payload needs one, it goes in the isolated Ollama-shaped sidecar behind a hard gate. That path stays available and it stays closed. The formal work here never needed the runtime anyway. The two mature Python libraries in this space are pure Python and optional, and MAX's exporter uses neither.

Where the reasoner would still win

MAX's entailment layer is a lint and a safe closure. It is not a complete classifier and it is not going to become one. I gave up two guarantees on purpose: satisfiability and subsumption checking over complex class expressions, and a proof that the ontology is globally consistent. A third loss is the justification trace, the explanation of why a given entailment holds.

Over crisp curated knowledge those are worth a great deal. Over this corpus they would be false comfort. A consistency proof across noisy contested data proves that the noise agrees with itself, and it proves nothing else. For MAX's workload the reasoner's strengths are the wrong strengths. That is a duller finding than a benchmark and a more durable one.

Two layers over one writer

MAX holds two kinds of knowledge with different shapes, so the design stopped trying to force one tool onto both.

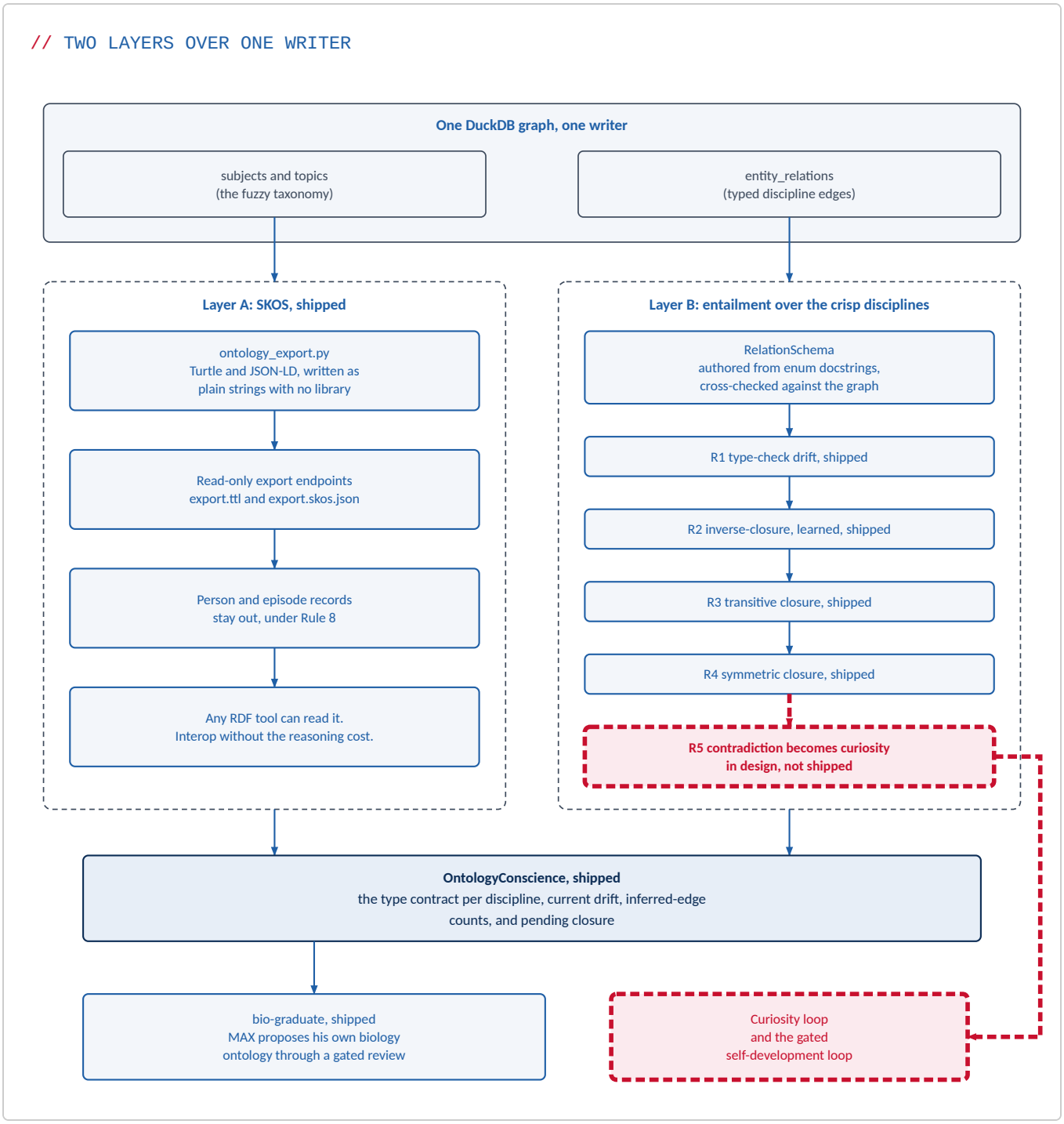


Figure 4. The red box at the foot of Layer B is the part that is designed and not shipped.

The fuzzy half is the discipline and subject taxonomy, and it gets the Simple Knowledge Organization System as a vocabulary and a read-only export. The exporter writes Turtle and JSON-LD, two standard serialisations of RDF data, and both come out as plain strings with no library dependency. Person and episode records stay out of the export under Rule 8. Any RDF tool can consume the result,

which buys the interoperability benefit of the standards without paying their reasoning cost.

The crisp half is the typed relations inside each discipline, where a relation has a real type contract. It gets five rule families written in Python.

RULE	WHAT IT DOES	HOW	STATUS
R1, type check	Confirms that a symptom_of edge really runs from a symptom to a disease	Schema verification against the graph, producing a drift list	Shipped
R2, inverse closure	Adds the missing reciprocal edge when a relation has a known inverse	A Forward-Forward network learns which relations are inverses, and edge maintenance applies them	Shipped
R3, transitive closure	If a relation holds from x to y and from y to z, it adds the edge from x to z	Schema closure over transitive candidates	Shipped
R4, symmetric closure	If a symmetric relation holds from x to y, it adds the edge from y to x	Schema closure over missing reciprocals	Shipped
R5, contradiction to curiosity	A cycle in an acyclic relation, a contraindication, or a disjoint clash raises a curiosity gap and penalises the weaker edge	Routes to a conflict record and an ontology-tension gap	In design

Every edge those rules apply is additive. A closure pass can add a missing reciprocal edge, and it can never downgrade a sourced edge at 0.9 confidence to an inferred one at 0.5. Applied edges carry capped confidence and an inferred provenance prefix, so nothing that came out of a rule can be mistaken for something that came out of a source. Everything is reversible and audited under Rule 9. Applying is off by

default and has to be armed. None of it sits on the voice path, which has a latency floor to protect under Rule 6.

The type contract is authored from the enum docstrings that already describe each relation, then cross-checked against the graph rather than trusted. That is the difference that matters against the Description Logic approach. Inverses are learned by a Forward-Forward network watching which relations turn up as pairs. Nobody declares them by hand in an axiom file.

COMPONENT	ROLE	STATUS
RelationSchema	The declared type contract for each discipline relation, authored from enum docstrings and cross-checked against real data	Shipped
Ontology lint detector	Runs the R1 type check and files propose-only schema-health gaps	Shipped
Schema closure	Applies the additive R3 and R4 closures	Shipped
Ontology export	SKOS as Turtle and JSON-LD, written without a library	Shipped
OntologyConscience	The unifying self-model: type contract per discipline, current drift, inferred-edge counts, and pending closure, exposed on its own endpoint and in the reasoning view	Shipped
Biology graduation	MAX proposes his own biology ontology through the gated path	Shipped
R5 routing	Live routing of a detected tension into curiosity	In design
Typed-tier cross-check	R1 over the dedicated junction tables that sit outside entity relations	Deferred
SKOS round trip	Import of an external ontology, against the current one-way export	Not planned

The conscience organ is the piece worth the attention. It gives MAX one view of the shape of his own knowledge and of the places where that shape disagrees with itself. The view carries the type contract per discipline, the current drift against it, the count of inferred edges, and the closure candidates still pending. The graduation path uses the same machinery to let him propose his own biology ontology through a gated review. The whole layer carries no AGPL, no Java runtime, and no second data store, which clears the commercial gate a future clinical product has to pass.

What is not built

WEAKNESS	THE HONEST READ
Not a complete classifier	There is no global consistency proof, no subsumption over complex class expressions, and no satisfiability checking. On a crisp curated domain a reasoner would beat this outright.
R5 is designed and not shipped	Type drift and additive closure run today. The contradiction-to-curiosity routing this whole arc is named for has a ticket number and no code behind it.
Only the entity tier is cross-checked	Typed relations in the dedicated junction tables are declared and not yet verified against the data.
The export runs one way	SKOS and OWL come out of the system. Nothing comes back in.
Applying is off by default	Closure is observable and has to be armed before it lands. That is safe, and it means the safe inferences wait on an operator.
No explanations	Nothing produces a justification in the Pellet sense. Provenance strings are the current substitute.

The headline of this arc is that a contradiction becomes curiosity. That routing is designed and it is not live. The first four rule families ship. The fifth sits in a design slice with a ticket number and no code behind it. Writing up an architecture whose

keystone is still on paper is how a design record turns into marketing. So the gap gets named here at the same size as the parts that work.

A Description Logic reasoner, handed the same two facts that disagree, would have been right to stop. It would also have been the last useful thing it did.

Glossary

TERM	DEFINITION
ABox	The assertion box, meaning the individual facts in an ontology, as distinct from the schema. A reasoner declaring an ABox inconsistent is saying those facts cannot all be true together.
AGPL	The Affero General Public License, a copyleft license whose obligations reach through a network boundary, so offering a modified program as a service can trigger a source-release requirement. It is Pellet's original license and the reason Pellet fails a commercial gate.
Closure	Adding the edges that already follow from the edges present, such as the missing third leg of a transitive chain. Safe closure means those additions are capped, marked, and reversible.
Decidable	A property of a logic guaranteeing that a correct answer arrives in finite time for every input. RDF Schema is decidable and cheap. The main Web Ontology Language dialect is decidable and expensive.
Description Logic	The family of formal logics underneath the Web Ontology Language, built for classifying concepts and checking whether a set of statements can hold at once.
DuckDB	An embedded analytical database that runs inside the host process rather than as a server. MAX keeps a single writer against it, which is the constraint a second knowledge store would have broken.
EL profile	A restricted subset of the Web Ontology Language that drops negation, disjunction, and universal restrictions in exchange for tractable reasoning. Large clinical terminologies are written to fit inside it.
ELK	A reasoner specialised for the tractable EL profile, running in polynomial time and in parallel. It classifies very large biomedical terminologies in seconds and cannot handle the constructs its profile leaves out.
Entailment	A fact that follows from the facts you stated, whether or not anybody wrote it down. Computing entailments is the job a reasoner exists to

TERM	DEFINITION
	do.
Epistemic ladder	MAX's ranking of how a claim was learned and how far it has been promoted, from a passing hint up to a well-supported belief. It is what lets two conflicting claims coexist at different weights.
Forward-Forward	A training method that replaces backpropagation with two forward passes, one over real data and one over generated contrast data. MAX uses it for the network that learns which relations are inverses of each other.
HermiT	A hypertableau reasoner from Oxford covering the full expressive dialect of the Web Ontology Language. It is efficient on many real ontologies and can exhaust memory on a hard one.
Java Virtual Machine	The runtime every production Description Logic reasoner requires. Adding one here would have meant a second garbage collector competing for memory with the language model and the voice path.
JSON-LD	A way of writing RDF data inside ordinary JSON, so web tooling can read it without a specialist parser.
Lint	A check that reports a problem and changes nothing. MAX's type-drift detector is a lint: it files a proposal and waits.
Monotonic	A property of a logic where adding a new fact never withdraws a conclusion already drawn. It suits curated knowledge and it fits corrected knowledge badly.
Open-world assumption	The rule that anything unstated is unknown rather than false. Application databases assume the opposite, which is why the Web Ontology Language surprises people arriving from one.
Openllet	The maintained open-source fork of Pellet under a permissive license, which is how the Pellet feature set can be used without the AGPL problem.
Pellet	A tableau reasoner covering the full dialect plus rules, datatype reasoning, and justifications. Its original license is AGPL and its

TERM	DEFINITION
	maintenance has moved to forks.
Provenance	The record of where a claim came from and how it arrived. Inferred edges in MAX carry a provenance prefix marking them as products of a rule rather than of a source.
RDF Schema	The lightweight schema layer for RDF data, supplying classes, subclass relationships, and property domains and ranges, with no negation, cardinality, or disjointness.
Satisfiability	Whether a concept can have any members at all without contradiction. An unsatisfiable concept usually means a modelling error.
Simple Knowledge Organization System	A W3C vocabulary for human taxonomies, offering concepts, a broader and narrower relation, an associative relation, and labels, with no logical entailment attached to any of them.
SNOMED CT	A clinical terminology of roughly 350,000 concepts, written inside the EL profile so a tractable reasoner can classify it.
Sound and complete	Sound means every entailment the engine reports is real. Complete means it reports every entailment there is. All three engines here are both.
SROIQ(D)	The specific Description Logic underneath the main Web Ontology Language dialect, naming the exact set of constructs the language allows.
Subsumption	One concept being a more general case of another. Classifying an ontology means computing all of these relationships.
TBox	The terminology box, meaning the schema and axioms of an ontology, as distinct from the individual facts.
Turtle	A compact plain-text syntax for writing RDF data, easier to read than the XML form and widely supported.

TERM	DEFINITION
Web Ontology Language	The W3C standard for expressive ontologies, supporting cardinality, disjointness, inverses, transitivity, and class expressions, and requiring a reasoner to produce value.

MAX3 WPR-MAX-016

// ABOUT THE AUTHOR

Jason Newell leads organizations and people in technology, and is a researcher, deep technologist and builder — from a CRAY to agentic AI. He founded MAX Research Collective to build AI for regulated industries that runs where its owner has control, with healthcare as its strongest focus. About Jason → Connect on LinkedIn →