

The Famulus Normal Form

Inferential relations between sentences, decided by joins in DuckDB

Gerd Graßhoff ^{ID}

Humboldt-Universität zu Berlin
gerd.grasshoff@hu-berlin.de

ABSTRACT

A scholar who holds one sentence and asks which sentences of a large edition entail it, contradict it, or merely sit near it, has no instrument that answers at the speed of a search. Language models answer, but at a price per pair, with errors that look like answers. This paper describes the Famulus Normal Form (FNF): a clause form into which every sentence of a collection is compiled once, so that the relation of any later probe to all of them is decided by joins in DuckDB and returned with a certificate. The organising idea is that inference is split from reading: a compiler reads the sentence, a dictionary normalises its concepts, rules add the literals that presupposition and implicature license, and the relation follows from which of the four truth-value cells are impossible. The reader will see the eight relation labels and their strength axis, the compile-normalise-match design, a sharded build over the Karman Nachlass, and a bake-off of the form against cross-encoders, an on-device model and cloud judges on a needle suite of 703 pairs. The claim is modest and exact: on short, regular sentences the form is the most precise classifier measured; on long historical sentences it recovers little, and the paper therefore places it in front of a language model judge, where it settles a quarter of the pairs on the measured subset without a measured loss of accuracy.

KEYWORDS natural language inference · DuckDB · normal form · entailment · contradiction · digital editions

1 The question and the failure of both easy answers

The question is a scholar's: which sentences of the edition stand in a logical relation to *this* sentence? A probe such as "The report was written in 1938" is asked of the whole Karman Nachlass, 663,972 sentences in English, German, French, Hungarian and Latin, and the answer must separate the sentence that entails it ("written in March 1938 in Pasadena") from the sentence that merely mentions a report and a year ("written in 1939"). The platform is index-first: an answer comes from a structure over pages, and generation only assists phrasing. A relation between sentences therefore needs a structure too.

Two easy answers exist, and each fails in a measurable way. A cross-encoder trained for natural language inference reads a pair in milliseconds, but it was trained on short sentences with three labels and it does not see the difference between *contradictory* and *contrary*; on the needle suite of this paper, of the pairs whose gold consequence is only implicated or defeasible, it calls 36 per cent strict consequences. A language model judge reads the pair with the full catalogue of relations in its instruction and reaches the best accuracy measured, but it is called once per pair: a probe against a million sentences is a million calls, and its errors arrive in the same fluent voice as its successes.

The third answer is the subject of this paper. The compiler is called once per *sentence*, not once per pair; the relation is then computed. That is cheap only if what the compiler returns is a form on which the relation can be decided by the database.

1.1 Contributions

1. **A relation catalogue with a strength axis** (Section 2): seven labels decided by four truth-value cells, an eighth label, *abductive*, decided by a weak test, and five strengths from necessary to defeasible.
2. **The Famulus Normal Form** (Section 3): a frame, a dictionary-normalised clause set, and the rules that derive presupposed, implicated and default literals from it.
3. **A matcher that is a join** (Section 4): literal entailment by sign, the four cells by clause cover, unit propagation and pair exclusion, in Python for verification and in DuckDB SQL for scale, with a certificate for each strict verdict.

4. **A sharded build and a haystack test** (Section 5): 593,840 sentences compiled in 16 shards, merged by Parquet, probed in SQL.
5. **A needle suite and a bake-off** (Section 6): 703 pairs in three slices, and five classifier families measured on accuracy, trap rate, speed and cost.
6. **A cascade and its limits** (Section 7): the form first, a cross-encoder and two language models behind it, and an honest account of what the form does not reach.

2 Relations, cells and strength

A relation label records how a stored sentence *S* stands to a probe *P*. It is read off four cells, the joint truth-value situations TT, TF, FT and FF of *S* and *P*; a label names the cells that are *impossible*.

Table 1 The seven strict labels and the cells that decide them.

Impossible cells	Label	Reading
TF	entails	whenever <i>S</i> is true, <i>P</i> is true
FT	entailed_by	whenever <i>P</i> is true, <i>S</i> is true
TF and FT	equivalent	each entails the other
TT and FF	contradictory	exactly one of them is true
TT	contrary	not both true, both may be false
FF	subcontrary	not both false, both may be true
none	independent	all four situations are possible

The eighth label is not truth-functional. *abductive* says that *S* is evidence and *P* the best explanation of it, under stated hidden assumptions, with a named defeater. It is decided in a second step, only when the strict label is *independent* or *entailed_by*: one pair can be strictly *independent* and weakly *abductive*. The catalogue (version 1.1.0) lists six conditions for it: *P* with the assumptions yields *S*; *S* does not entail *P*; both can be true; every rival explanation is excluded or clearly worse; a defeater exists; and the link runs through the content of *S*, not through a shared topic.

CONCEPT Strength is not confidence

A verdict has a **strength**: how firmly *P* follows from *S*. The axis runs necessary, presupposed, hedged, implicated, defeasible. It is orthogonal to the label: “the experiment was scheduled for February” gives *independent* strictly and *entails* weakly, at strength defeasible, because scheduled events usually take place as scheduled. A verdict is as strong as the weakest rule or dictionary edge it used. The axis exists so that a defeasible consequence is never reported in the voice of a necessary one.

Two test families keep the axis honest. A *defeater pair* adds a clause that cancels a weak consequence (“the experiment was postponed”); the weak verdict must flip, and a necessary verdict must stay, because a necessary verdict that flips is a monotonicity bug. A *twin* changes one fact that a strict relation must be sensitive to (1938 to 1939) and the verdict must change with it.

3 The Famulus Normal Form

The form has four layers, and the division between them is the design.

Table 2 The layers of the Famulus Normal Form (FNF).

Layer	Content	Producer	Stored in
L0	the text	the corpus	fnf_sentence
L1	the frame: propositions with roles and a conjunctive normal form over them	UD compiler, LLM compiler, ensemble	frame JSON
L2	normalised atoms and literals: concept keys, canonical orientation, distributed plurals, derived literals	the normaliser, deterministic	fnf_atom, fnf_atom_role, fnf_lit
L3	the same, indexed for joins	the indexer	DuckDB

3.1 The frame

A *frame* is the rule-free and dictionary-free reading of one sentence. Each proposition carries a predicate, a role table, a modality (`assert`, `probable`, `possible`, `necessary`, `reported`, `desired`, `hypothetical`, `embedded`), an optional holder and a plural reading. The `cnf` field is a list of clauses over the propositions, so that “A or B” and “not A” have a place. Every predicate and value is an English lemma whatever the source language: the frame is an interlingua, and a German sentence and its English translation should compile to the same frame. Dates become interval rows (`time.year`, `time.month`, `time.day`), numbers become amount intervals, and names are reduced to entity keys with titles, particles and accents removed.

Two compilers produce frames. The *UD compiler* reads the Universal Dependencies parses already stored for the collection and applies function-word tables for English, German and French; it is deterministic and fast, and it reports a confidence. The *LLM compiler* sends the sentence with the frame prompt and the dictionary vocabulary to a model and validates the answer against the frame schema; a failed validation is retried once with the problems named, and results are cached by sentence, model and prompt hash. An *ensemble* runs the UD compiler first and calls a model only when the confidence is below a threshold.

3.2 The dictionary

The dictionary maps lemmas of five languages to concept keys and relates the keys by typed edges: `isa`, `postulate`, `complementary` and `scalar antonyms`, `disjointness` and `converse`. Each edge has a strength (N or D) and an evidence grade: `DOCUMENTED`, `INFERRED` or `HYPOTHESIS`. The closure `concept_up` is materialised in DuckDB. The certified tier uses necessary edges of grade `DOCUMENTED` only. A WordNet import adds first-sense hypernym chains for the most frequent noun and verb lemmas of the corpus; its keys are synset keys (`wn:document.n.01`) so that the senses of a polysemous lemma cannot merge, and every edge is graded `INFERRED` and so stays out of the certified tier.

3.3 Normalisation and derived literals

The normaliser is deterministic and re-derivable from L1. It maps values to concepts, rewrites converse predicates to the canonical orientation with agent and theme swapped, rewrites complementary antonyms to the canonical polarity (so that `unmarried(x)` is `not married(x)`), distributes distributive plurals into one atom per member, and joins collective plurals into a single `set` value. Then it derives literals in two passes.

The first pass adds literals of strength necessary and presupposed: definitions (`unmarried` and `man` give `bachelor`), implicative verbs (to manage to copy entails copying; to fail to copy entails not copying) and presuppositions, which project through negation and through *perhaps*. The second pass adds literals of strength implicated and defeasible, and the abductive candidates. Each is dropped when a first-pass literal

contradicts it, when a named defeater occurs in the sentence, or, for an implicature, when the sentence makes the stronger alternative possible.

A default rule

```
- id: D.scheduled_occurs
  kind: default 1
  grade: D
  when: [{pred: c:schedule, roles: {theme: ?x, time: ?t}}] 2
  derive: {pred: c:take_place, roles: {theme: ?x, time: ?t}}
  assumptions: [{grade: standard, text: scheduled events usually take place as
scheduled}]
  defeaters: [c:postpone, c:cancel] 3
```

- 1 The kind fixes the pass and the strength grade.
- 2 The pattern binds the event and its time.
- 3 A positive atom with a defeater predicate removes the derived literal.

The rule is a datum, not code: the rules live in a YAML file, each with its hidden assumptions and defeaters, and a rule pair generator turns every rule into minimal test pairs in three languages. A default or abductive rule also copies the time, place and amount rows of its trigger onto the derived atom, because an explanation of an event holds in the circumstances of the event; without that copy, “the flow separated above 16 degrees” does not follow from a derived `separate(flow)` that has lost the 16 degrees.

4 Matching

4.1 Literal entailment

Matching starts from literals. For a stored literal m and a probe literal l the entailment $m \sqsupseteq l$ holds by sign:

Table 3 Literal entailment by sign.

m	l	holds if
positive atom a	positive atom b	a subsumes b : predicate and modality up the dictionary, same holder, every row of b entailed by a row of a
negative a	negative b	b subsumes a
positive a	negative b	a excludes b : antonym or disjoint predicates, or a clash on a functional role (<code>time.*</code> , <code>place</code>) in one event
negative a	positive b	complementary antonyms with identical participants

4.2 The four cells

With literal entailment fixed, the four cells are clause-set questions. TF is impossible if every probe clause is covered by a stored clause, with disjunctive syllogism. FT is impossible by the same test in the other direction. TT is impossible if unit propagation over both clause sets reaches the empty clause. FF is impossible if every pair of negated base clauses is inconsistent. The impossible set is mapped to the label of Table 1. The weak

pass runs afterwards, only for `independent` and `subcontrary`, with the literals of grades I, D and A added; the strongest grade that the weak trail used decides between a weak `entails` and `abductive`.

A strict verdict carries a certificate: the cells that are impossible, for each the trail of literal pairs, the dictionary edges used, and, for a weak verdict, the rules, the hidden assumptions and the defeaters. The certificate is what makes a fast answer accountable.

METHOD Integrity rule for the form

The layers above the frame never lag it. L2 is re-derived from L1 whenever the dictionary or the rules change, without recompiling the corpus, and a strict verdict never depends on a literal of grade I, D or A. A frame that fails schema validation, including one whose role values refer to a proposition that does not exist, is treated as not compiled and never as an empty sentence.

5 Inside DuckDB

5.1 Tables and the SQL matcher

The index has four tables: `fnf_sentence`, `fnf_atom` (key, predicate, modality, holder, row count), `fnf_atom_role` (one row per role value with its interval) and `fnf_lit` (sentence, clause, sign, grade, rule). The dictionary adds `concept`, `lemma_concept`, `concept_edge`, `concept_up`, `concept_antonym`, `concept_excl` and `mod_up`. A probe is loaded into three temporary tables of the same shape. The matcher computes atom pairs that entail each other by joining the two atom tables through `concept_up` and the role rows, aggregates by sentence with a `HAVING` on clause cover, and returns one row per candidate with the four cell flags and a flag `exact`. The SQL is exact when every clause of both sets is a unit; otherwise `exact` is false and the Python matcher verifies. A parity test runs both matchers over the same frames.

Sentences that share no literal with the probe are independent by construction and are not returned. A bare predicate is not a statement: “COPY” or “IN REPLY”, a fragment that the parser cut from a letterhead, is entailed by any probe that uses the predicate and is subcontrary to its negation. The matcher therefore takes an informativeness gate, `min_rows`, that drops sentences none of whose atoms has a role row; on the haystack below the gate removes 2,574 of 2,576 hits.

5.2 A sharded build

The corpus build splits by document hash, so that a document is never divided, compiles each shard in its own process, and writes Parquet per shard through a temporary directory and an atomic rename; a shard that exists is skipped, so the build resumes. The merge is one `insert ... select distinct on (atom_id)` per table over a list of Parquet files. In the measured build the UD compiler reached its confidence threshold for 180,559 of 593,840 English, German and French sentences, 30 per cent; the other 413,281 are queued for the LLM stage, and Hungarian and Latin, for which no function-word table exists, wait there too. The tables are written once more, as a single snapshot, into a DuckLake schema, which is the form in which the platform’s other layers are served.

Confidence is not agreement. On 200 UD-confident sentences per language the LLM compiler’s frame and the UD frame share on average 8.8 per cent of their normalised clauses for English and none for German and French, because the UD compiler keeps source-language lemmas where the LLM compiler writes English ones, and because many corpus sentences are fragments cut from letterheads and typescript, which the UD rules compile into propositions that a reader would not accept. The UD output is therefore used for what it is good at: a cheap, deterministic hay of 180,559 sentences against which the join is timed, and a screen that tells which sentences need no model call. A relation claimed between a probe and a UD-compiled fragment is not evidence; a relation claimed between a probe and an LLM-compiled sentence is, to the degree that the compilation is.

Table 4 Throughput of the build from UD parse to index, one process. Compile is the UD compiler without the confidence threshold, so every sentence is compiled; match is one SQL pass for one probe.

Sentences	Compiled (UD)	Literals	Com- pile+nor- malise (s)	Index (s)	Match, me- dian per probe (s)	Sentences/ s (compile)
1,000	1,000	1,264	0.5	0.04	0.14	1,946
10,000	9,966	11,739	3.5	0.14	0.151	2,869
100,000	99,951	114,816	46.2	1.57	0.195	2,163
593,840	592,594	688,840	113.1	5.87	0.424	5,250

5.3 A haystack test

The needle suite tests the relations; the haystack tests the scale. The sentences of 103 needle pairs of the seeds and rule slices are injected into a hay of real, UD-compiled corpus sentences, each probe is matched against hay and needles in one SQL pass, and the injected sentence must come back with its gold strict label.

Table 5 Needles in a haystack of real, UD-compiled corpus sentences. The same 103 needle pairs at every size.

Hay	(sen- tences)	Literals	Needle recall (%)	Wrong label (%)	Hay max	hits, SQL per per probe probe	seconds probe (median)
1,000		1,283	75.7	2.9		0	0.055
10,000		11,285	75.7	2.9		0	0.0555
100,000		110,255	75.7	2.9		1	0.0959

The recall for the needles is the same at every hay size, 75.7 per cent, because it is limited by the compiler agreeing with itself on the pair, not by the join; a repeat at the largest size returned 76.7 per cent, one needle different. The cost grows slowly with the hay. After the informativeness gate at most one hay sentence per probe is reported as related to the probe at any size; the one hay hit at the largest size was read and is wrong: a fragment, “will not happen and it was concluded that the addition of weight and”, compiled as the negation of an event and returned as contrary to a probe about an experiment in March 1937. The false-positive rate of the matcher on real sentences is therefore small but not zero.

OPERATION A probe against the collection 0.10 s

Q Which sentences of the Karman Nachlass entail “Karman wrote a letter.”?

- fnf.compile.llm** compiles the probe to a frame and validates it against the schema
- fnf.normalise** maps the frame to concept-keyed literals and derives the presupposed and implicated ones
- fnf.match_sql** joins the probe tables against the literal table of the index and returns the four cell flags per sentence
- fnf.infer** drops fragments, attaches the **fr:** address of each hit and returns the hits with their labels

→ the sentences whose label is **entails**, **equivalent** or **entailed_by**, each with its page address

The operation returns in about a tenth of a second of SQL over a hay of 100,000 sentences after the probe is compiled; the compile call is a single model request. The function `fr_infer` wraps it; its descriptor is ready for registration on the famulus-reference server, which is not part of this repository, and the registration is therefore still open.

■ 6 Evaluation

■ 6.1 The needle suite

The needle suite has 703 pairs in three slices. The *catalog* slice has 467 pairs built from real sentences of the Karman, Hevelius and Kepler benchmark, with a probe generated for each relation. The *seeds* slice has 157 pairs from hand-designed families: necessary, presupposition, hedging, implicature, defeasible, generic, abductive, direction, causal-temporal and cross-sentence. The *rule* slice has 79 pairs: the rule pair generator verbalised every rule in English, German and French, a second, stronger model judged each pair, and only the 79 on which the judge reached the gold strict and weak label were kept; the other 185 are kept apart as disputed. Each pair also carries variants (a noisy transcription, twins, a defeater), so that the 703 pairs yield 1,505 judgements.

The gold is generated and partly judge-agreed. It is not human-verified, several families have fewer than thirty items, and the seed families were written alongside the rules, so the figures for the seeds slice are not independent evidence for the rules. The catalog slice is the one that the rules did not see.

■ 6.2 The bake-off

Five classifier families were run on the same suite with the same instruction where one is used: the form compiled by two models, a cross-encoder (mDeBERTa-v3 trained for cross-lingual inference, both directions), the on-device Apple Foundation Model through a closed-choice helper, and cloud judges that read the full catalogue and answer with four cells and, in the weak variant, a weak label. A reduced subset of 160 pairs, drawn with a fixed seed, is the common ground for the slow models.

Table 6 Bake-off on the needle suite (a cascade is named form › cross-encoder › fast judge › strong judge). Figures other than Items, Pairs/s and USD are per cent. Strict is strict accuracy, the share of pairs with the gold strict label; catalog, seeds and rule are slice accuracies in per cent; subset is the common subset, the 160 pairs that every run, including the slow models, scored. The trap rate is the share of close-but-negative pairs called related; over-entailment is the share of pairs without support called entailed. Pairs per second are wall-clock on a loaded machine with the workers each family used and are not comparable across families; cost (USD) is estimated from token counts. Abst. is the share of pairs without a usable answer; Trap and Over-ent. are the trap rate and over-entailment.

Classifier	Items	Strict	Abst.	Sub-set	Cata-log	Seeds	Rule	Trap	Over-ent.	Pairs/s	USD
cascade flash-lite › nli › haiku-5-5 › 3.8-flash	160	86.3	0.0	86.3	82.7	92.9	95.5	3.8	4.0	0.22	–
judge haiku-5-5	703	84.4	0.0	86.3	85.0	85.4	78.5	5.6	3.5	0.51	11.49
judge 3.8-flash	160	82.5	0.0	82.5	80.9	82.1	90.9	0.0	4.0	0.17	1.41
cascade flash-lite › nli › flash-lite › 3.8-flash	703	69.6	0.0	71.3	60.4	89.2	84.8	23.8	8.0	0.45	–
judge deepseek-v4.1-flash	160	68.8	25.6	68.8	69.1	71.4	63.6	1.9	4.0	0.21	–
judge glm-5.3-flash	160	60.0	37.5	60.0	53.6	71.4	77.3	1.9	4.0	0.12	–
judge flash-lite	703	59.5	0.0	55.0	55.7	71.3	58.2	24.6	14.2	2.25	0.52
FNF flash-lite	703	52.6	0.3	50.0	34.3	91.1	84.8	0.4	0.0	2.85	0.57
cross-encoder mDeBERTa	703	47.5	0.0	46.3	44.3	63.7	34.2	28.6	36.3	7.17	–
FNF haiku-5-5	703	43.2	31.2	44.4	20.3	90.4	84.8	0.0	0.0	0.27	19.24
Apple AFM (on-device)	703	31.9	0.9	33.1	26.1	41.4	46.8	27.4	72.6	0.93	–

Three results stand out. The form is by far the most precise classifier: it reaches 91.1 per cent on the seeds slice and 84.8 on the rule slice, with a trap rate of 0.4 per cent and no over-entailment, where the trap rate is the share of close-but-negative pairs that a classifier calls related. On the catalog slice it reaches 34.3 per cent: it answers `independent` for 96 per cent of the real sentence pairs. The strongest single judge measured is `claude-haiku-5-5`: 84.4 per cent on the full suite, 85.0 on the catalog, flat across sentence length, at an estimated USD 11.49 and 0.51 pairs per second. `gemini-3.8-flash` reaches 82.5 per cent on the 160-pair subset where `claude-haiku-5-5` reaches 86.3, at USD 1.41. `gemini-3.5-flash-lite` is about four times faster and twenty times cheaper than `claude-haiku-5-5` and far less careful on trap pairs (a trap rate of 24.6 against 5.6 per cent). Two further cloud judges, `deepseek-v4.1-flash` and `glm-5.3-flash`, are accurate when they answer (92.4 and 96.0 per cent) and returned no usable answer for 25.6 and 37.5 per cent of the 160 pairs; the cause was not traced, and a direct test of nine pairs afterwards returned no error, so the abstentions may belong to the load of the run rather than to the models. The form compiled by `claude-haiku-5-5` abstained on 31.2 per cent of the pairs because its frames did not validate. The cross-encoder and the on-device model are fast and cheap and unreliable: the cross-encoder cannot separate `contrary` from `contradictory`, which accounts for 53 of its catalog errors, and the on-device model entails almost everything it is shown.

Table 7 How each classifier is wrong, per cent of its items. Missed: gold names a relation and the classifier says independent. Invented: gold is independent and the classifier names a relation. Mislabelled: both name a relation, not the same one.

Classifier	Items	Correct	Missed	Invented	Mislabelled	Abstained
cas-cade[gemini-3.5-flash-lite]	nli	claude-haiku-5-5	gemini-3.8-flash	160	86.2	3.1
judge claude-haiku-5-5	703	84.4	5.7	4.8	5.1	0.0
judge gemini-3.8-flash	160	82.5	11.2	3.8	2.5	0.0
cas-cade[gemini-3.5-flash-lite]	nli	gemini-3.5-flash-lite	gemini-3.8-flash	703	69.6	3.4
judge deepseek-v4.1-flash:cloud	160	68.8	1.9	2.5	1.2	25.6
judge glm-5.3-flash:cloud	160	60.0	0.6	1.9	0.0	37.5
judge gemini-3.5-flash-lite	703	59.5	4.1	14.2	22.2	0.0
FNF gemini-3.5-flash-lite	703	52.6	45.9	0.3	0.9	0.3
cross-encoder mDeBERTa	703	47.5	3.7	22.9	25.9	0.0
FNF claude-haiku-5-5	703	43.2	22.0	0.7	2.8	31.2
Apple AFM (on-device)	703	31.9	0.0	35.1	32.1	0.9

The error table separates two kinds of mistake that the accuracy hides. The form’s errors are almost all misses: 45.9 per cent of its pairs are pairs with a gold relation that it calls `independent`, and 0.3 per cent are relations it invents. The two small judges and the on-device model err the other way: `gemini-3.5-flash-lite` invents a relation for 14.2 per cent of the pairs, the cross-encoder for 22.9 and the on-device model for 35.1. A miss costs a call to a judge in the cascade; an invented relation costs a wrong answer that nobody checks.

Table 8 Strict accuracy (per cent) by the length of the stored sentence S.

Classifier	≤10 words	11–20	21–35	>35
cascade flash-lite › nli › haiku-5-5 › 3.8-flash	92.6	66.7	88.7	74.1
judge haiku-5-5	83.4	84.2	86.9	82.1
judge 3.8-flash	86.8	75.0	83.0	74.1
cascade flash-lite › nli › flash-lite › 3.8- flash	84.5	64.4	58.7	54.7
judge deepseek- v4.1-flash	69.1	75.0	73.6	55.6
judge glm-5.3- flash	69.1	50.0	58.5	44.4
judge flash-lite	66.2	54.5	56.8	50.9
FNF flash-lite	74.5	43.6	33.5	38.7
cross-encoder mDeBERTa	52.1	44.6	48.1	36.8
FNF haiku-5-5	74.8	43.6	15.5	10.4
Apple AFM (on- device)	42.1	21.8	27.2	22.6

The length table gives the reason for the form’s catalog result. Its accuracy is highest for sentences of ten words or fewer (74.5 per cent) and falls to 33.5 per cent for sentences of twenty-one to thirty-five words, where the judges stay flat. Inspection of the errors shows the mechanism: a long Hevelius sentence with a conditional compiles to a different clause structure from its own paraphrase, and a relation of the form is decided on exact structure, so one divergent clause removes the relation. The error is a loss of recall, not of precision, and this is why the form is placed in front of the judges rather than instead of them.

6.3 Weak consequences

Table 9 The weak tier. Weak accuracy scores the pair against the gold weak label; defeater flip is the share of weak verdicts that flipped when a defeater was added; a monotonicity violation is a necessary verdict that flipped.

Classifier	Weak acc. (%)	Defeater flip (%)	Monotonicity violations (%)	Abduction recall (%)
cascade flash-lite > nli > haiku-5-5 > 3.8-flash	81.2	78.6	8.7	50.0
judge haiku-5-5	81.1	87.7	19.7	44.4
judge 3.8-flash	80.0	83.3	11.1	37.5
cascade flash-lite > nli > flash-lite > 3.8-flash	66.0	75.3	5.5	63.9
judge deepseek-v4.1-flash	65.0	75.0	13.3	12.5
judge glm-5.3-flash	57.5	83.3	14.3	25.0
judge flash-lite	53.5	57.7	31.4	66.7
FNF flash-lite	42.0	56.0	7.7	16.7
cross-encoder mDeBERTa	45.2	50.0	23.5	0.0
FNF haiku-5-5	33.3	63.3	25.6	22.2
Apple AFM (on-device)	39.4	44.0	9.8	0.0

The weak tier is the weakest part of the form. Abduction recall is low because the abductive rules fire only when the compiler has chosen the role structure that the rule pattern names; the form recovers the weak label of 26.8 per cent of the 71 pairs with a defeasible or implicated consequence and of 16.7 per cent of the 36 abductive pairs, and it never supports an unsupported pair. A defeater flips the weak verdict in 56 per cent of the 25 pairs that have one. The necessary verdicts of the form violate monotonicity in 7.7 per cent of the 78 pairs that have a defeater variant: six pairs (three German, two French, one Latin) whose cause has not been traced pair by pair. These figures are reported as measured; the strict tier does not depend on the weak tier.

7 The cascade

The measurements suggest an order of use rather than a winner. The form is cheap per pair, certified and precise but silent on long sentences; the judge reads everything and is expensive and sometimes wrong in a fluent voice. The cascade runs them in that order. Tier one is the form: a decisive strict label, or a weak label, is final. Tier two is the cross-encoder, which may close a pair as independent only if it sees no signal in either direction and the two sentences share no content word. Tier three is a fast language model judge. Tier four is a strong judge, called only when the fast judge and the cross-encoder disagree on whether any strict relation holds, or when the fast judge failed.

Table 10 The cascade on the needle suite. The last four columns are the per cent of pairs decided at each tier.

Cas- cade (form / fast / strong model)	Strict acc. (%)	Cata- log	Seeds	Rule	Trap rate (%)	Form	Cross- enc.	Fast	Strong
gem- ini-3.5- flash- lite / claude-haiku-5-5 / gem- ini-3.8- flash	86.3	82.7	92.9	95.5	3.8	25	1	47	26
gem- ini-3.5- flash- lite / gem- ini-3.5- flash- lite / gem- ini-3.8- flash	69.6	60.4	89.2	84.8	23.8	28	2	47	23

The routing statistics are the quantity the cascade is tuned on. In the run with `claude-haiku-5-5` as the fast tier, on the 160-pair subset, 25 per cent of the pairs were decided by the form, 1 per cent by the cross-encoder, 47 per cent by the fast judge and 26 per cent by the strong judge. The cross-encoder tier closes almost nothing, because content-word overlap is nearly always present between two sentences of one edition; it is the least useful tier and the first candidate for removal. The accuracy of the cascade is bounded by its fast judge: with `gemma-3.5-flash-lite` as the fast tier it reaches 69.6 per cent on the full suite, with a trap rate of 23.8 per cent, and with `claude-haiku-5-5` it reaches 86.3 per cent on the 160-pair subset, with a trap rate of 3.8 per cent. On that subset the `claude-haiku-5-5` judge alone also reaches 86.3 per cent. The cascade with the stronger fast tier therefore saves a quarter of the judge calls without a measured loss of accuracy, at a lower pair rate (0.22 against 0.51 per second, because the strong tier is slow); it does not gain accuracy, and the two trap rates are measured on different item sets and are not compared.

8 What the form does not do

The limits are specific. Gold is generated: the 703 pairs have no human verification, and a human review queue exists but has not been signed. Hungarian and Latin sentences have no UD compiler and wait for a model; the UD compiler is confident for under a third of the English, German and French sentences, and its frames agree with the LLM compiler’s on few clauses, so the 180,559 UD-compiled sentences are a hay, not a compiled collection. The SQL matcher is exact only for unit clause sets. The dictionary is a seed of about ninety hand-written concepts and a first-sense WordNet import whose edges are inferred; the rules are thirty-six, written for the families that the suite tests, and the rule slice measures them against pairs derived from themselves. The abductive tier recovers a minority of its pairs. The form decides what the compiler gave it: a sentence compiled with the wrong predicate has the right relation to the wrong sentence.

■ 9 Conclusion

The Famulus Normal Form moves the work of inference from the pair to the sentence. A sentence is compiled once into concept-keyed clauses, and every later probe is a join that returns a label with a certificate. In the measured hay of 100,000 sentences the join returns in about a tenth of a second, and the needles are found at every hay size at the same rate. The form does not replace a language model on long historical sentences; it removes from the model the pairs that a certificate can settle and refuses to answer the others.

Three pieces of work are planned, each with an owner in the repository. The first is a human review of the 703 gold labels, starting from the review queue that the suite builder writes. The second is the decomposition of long sentences into clause-sized frames before matching, which is the measured remedy for the loss of recall on long sentences. The third is the LLM stage of the sharded build for the 413,281 pending sentences, followed by the registration of `fr_infer` on the famulus-reference server.