

The Famulus Probe Lab

Normalise, filter, classify: generating critical annotation for the items of a reference storage

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

ABSTRACT

A reference storage that gives every page and segment of a scholarly edition a stable address, a citation and a provenance chain still leaves the reader alone with the sentence. A letter says “he”, “the office”, “last Thursday”; a scholar who holds one claim and asks which sentences of the edition entail it, contradict it, or merely stand near it, cannot compare the claim with such sentences, because neither side is yet a statement with fixed reference. This paper describes the Famulus Probe Lab, a working bench that turns a probe, a sentence of the scholar, into critical annotation of the stored items in three stages. Normalise reads the probe and the sentences with the apparatus of an edition: every deictic, nominal, temporal and anaphoric expression receives a footnote with grounds, assumptions, objections and a grade, under the rule that no value stands without a ground, and the result is a non-destructive, versioned XML normal form and a set of justified numeric keys. Filter turns the keys of the probe into one bound SQL query and reduces the sample, here from 200 sentences to 4. Classify, which belongs to the companion benchmark famulus-logic, decides for each survivor whether it entails the probe, contradicts it, is independent of it (logically irrelevant), or explains it abductively, through a four-tier cascade, followed by a cross-sentence abduction and a three-sentence support test. Each stage has its own declarative rule files, its own variants and a gated learning loop in which an instruction becomes edits of those files, is tested and is run against its basis. The paper is conceptual: it states the annotation model, the invariants that make an annotation critical rather than merely descriptive, and the division of labour between stages. It reports a single worked case, one probe against a sample of 200 sentences of the Kármán Nachlass, and says plainly what that case does and does not show.

KEYWORDS critical annotation · reference storage · entailment · contradiction · abduction · DuckDB · digital editions · famulus

1 From a reference storage to a critical annotation

A reference storage is a promise about citation. In the Famulus reference architecture (famulus-reference/1) every document, page and segment of a collection has an `fr:` address of the form `fr:<collection>/<doc>/<pid>/<segment>`; every address resolves to a text layer with its hash, to a citation that is taken only from recorded bibliographic data, to the steps and tools that produced the text, and to the validation record of those steps; and every answer names the snapshot of the DuckLake from which it was read. A citation that no ancestor of the item carries is returned as a gap and is never composed from a title or a file name. The storage is thus careful about *where* a text is and *where it came from*. It says nothing about *what the text says*.

That second question is the one a scholar asks. The scholar holds a claim, here called the *probe*, and wants the sentences of the edition that bear on it: those that entail it, those that contradict it, those that explain it, and, as important, those that only look as if they did. Two obstacles stand in the way, and they are different in kind.

The first is *reference*. A letter of the Kármán Nachlass says “He went to St. Louis and returned to Pasadena.” The sentence cannot be compared with a probe about the office in St. Louis until “He” is settled, and whether it can be settled depends on the letter head, on the neighbouring letters of the exchange, and on a register of persons. A relative date (“last Thursday”) needs the day of writing; “the office” needs an owner. Where the edition’s text does not decide, a reader of an edition does not guess: she writes a footnote that names the

ground, the assumption and the alternative that was not chosen, and where nothing grounds a reading she says so.

The second is *relation*. Even when both sentences are statements with fixed reference, the question which relation holds between them is not answered by thematic similarity. A sentence that shares the place and the year with the probe may be independent of it; a sentence that shares nothing but a rule of meaning may entail it. Language models answer the question fluently and at a price per pair, and their errors have the same voice as their successes.

The Famulus Probe Lab is the bench on which the answer to both obstacles is built and tested. It is an implementation, `scripts/satz_labor.py` of the `famulus-proof` library together with the classifiers of the benchmark `famulus-logic`, and this paper uses it as the occasion to state a conceptual model: how *critical annotation* of the items of a reference storage can be generated, so that it is addressed, graded, reversible and open to objection.

1.1 What is meant by critical annotation

We use the word in the sense of an edition. An annotation is critical if it satisfies five conditions, each of which the lab enforces by a mechanism and not by good intentions.

1. **It quotes.** The annotation is attached to a span of the item, a range of tokens and characters, and the span is the quoted phrase.
2. **It grounds.** The annotation lists the reasons that carry its value, each with an address at which the fact can be checked.
3. **It is graded and defeasible.** The grade is computed from the reasons, never entered by hand, and a defeater lowers it or withdraws the value.
4. **It keeps the alternatives and the gaps.** The readings that were not chosen are recorded, and a reading that nothing grounds stays open, with the reason it is open.
5. **It never destroys the item.** The item is returned exactly by removing the annotation, and an annotation is never overwritten: a change is a new version with a reason.

A critical annotation in this sense has two layers. The *expression layer* annotates single spans of a sentence (what “his” refers to, which place “St. Louis” is, what the tense and polarity of the predicate are). The *relation layer* annotates the pair of a stored sentence and the probe (entails, contradicts, is independent of, explains) with a strength and a certificate. The lab produces both.

1.2 Contributions

1. A model of critical annotation for the items of a reference storage in two layers, with five conditions and one invariant that carries them: *no value without ground* (Section 2).
2. The nine-stage normalisation of expressions, its type theory, its guards and its grade function, and the non-destructive XML normal form it writes (Section 3).
3. A filter that is a function of the justified keys of the probe alone: anchors, one bound query, an error matrix (Section 4).
4. A classification of the survivors by the four-cell relation catalogue of `famulus-logic`, with entailment, contradiction and independence as its three regimes, a weak test and a strength axis, a cascade, and two inferences over more than one sentence (Section 5).
5. A gated A/B learning loop for the declarative rule files of each stage, and the record of the loop on the entry case (Section 6).

2 The annotation model over a reference storage

2.1 Items, sidecars and the normal form

The lab works on three stores, and the distinction between them is the first design decision.

Table 1 The three stores of the lab. The reference storage is the schema `famulus-reference/1`.

Store	Holds	Written by	Never
Reference storage	addresses, text layers, citations, provenance, validation	the intake pipeline	edited by the lab
Sentence sidecar <code>{coll}_satz.duckdb</code>	sentences with Stanza tokens, clauses, keys, attributes, footnotes and their reasons	the sidecar build and the normalise stage	a copy of the truth; it is rebuilt
Version store <code>{coll}_normal.duckdb</code>	the XML normal form of each letter, version by version, curator additions, inquiries	the normalise stage and the curator	overwritten

The sentence sidecar is the working surface. Its tables are `sr_sentence` (the sentences with their Stanza analysis), `sr_clause` (clauses cut from the dependency tree), `sr_phrase` (the phrases of the normalise stage), `sr_expr` and `sr_expr_grund` (the footnotes and their reasons), `sr_key` and `sr_attr` (the keys). A key is the 64-bit hash of a kind and a canonical value, `hash(kind || chr(31) || canon)`, so it is deterministic and a rebuild changes none. The comparison of two sentences by keys is therefore a comparison of numbers, and the only place where a surface form meets the register is the step that classifies it.

The sidecar addresses a sentence as `S:<doc>#<pid>.<sentence>`, for example `S:1-13_alperin_morton_1945_1955#18.10`. This address is built from the document and the paragraph index that the `fr:` address of the page and segment also carries. We treat the sentence address as the sidecar's local address for an item that lies below a segment of the storage, and we propose, but have not built, an explicit mapping from it to an `fr:` address. This matters for the conceptual claim: the annotations the lab produces are annotations of items of the storage only to the degree that this mapping is fixed. At present the lab keeps them next to the sidecar, not in the DuckLake.

2.2 The normal form

The expression layer is written into the letter itself, in a form that cannot lose it. The normal form of a letter is the letter's XML with each annotated phrase wrapped in an `expr` element and a `normalisation` back block that explains every wrapper. A phrase is a range of tokens and characters of a sentence, and the sentence is located in the letter's XML; a mark is placed around exactly that range. Where an element boundary of the letter cuts the range, the mark is split into parts at the boundary, each part pointing to the first with a `teil` attribute. Marks nest by their ranges: the place "St. Louis" lies inside the unit "the St. Louis office" and is marked inside it. Removing all marks and the back block returns the original XML exactly, and the implementation tests that `(entfernen(annotieren(xml)) == xml)`.

The normal form has three views, all computed from the same XML: `original`; `normal`, in which marked phrases are replaced by their normal form, with each open footnote rendered as an explicit gap; and `interlinear`, which keeps the original and adds the normal form beside it. The `normal` view is what the classifier reads in the third stage.

A letter is stored in the version store as a sequence of versions. A version is never changed or deleted; a later version carries a reason and points to its predecessor, and the kind of change is one of five.

Table 2 Kinds of change (art) of the normal form.

Kind	Meaning
lauf	a plain run of the normalise stages
kuration	a curator, or an inquiry, added a person, an owner, a referent, a place or a note
regelaenderung	the hash of the rule files changed
kontextaenderung	a letter on which the version rests has a newer version
erkundung	an inquiry grounded or refuted a background reading

A version is *stale*, and the store says so without redoing anything, when the rule files differ from those it was made with, when a letter it depends on has a newer version, or when an addition was made after it. Nothing is redone silently.

CONCEPT Annotation as a versioned object

An annotation of an item is not a field of the item. It is a version of a derived document, produced by named rule files whose hash is recorded, from a named context whose hash is recorded, with a reason when it replaces another version. Whoever later asks why a reading was made can reproduce the version, and whoever changes a rule can ask which versions that change makes stale.

2.3 The invariant: no value without ground

The four conditions of grounding, grade, alternatives and non-destruction reduce to one invariant that the program checks. A footnote that has a value must have at least one *carrying* reason with a grade and an address; a footnote with no carrying reason has no value and is open. A reason is one of ten kinds, and of these six carry: *faktor* (a fact of the syntactic analysis), *kontext* (the letter record), *register* (the register of individuals), *serie* (another letter of the exchange), *beleg* (a piece of evidence found by an inquiry) and *kuration* (a curator's statement). The others qualify: *annahme* (an assumption), *waechter* (a defeater), *alternative* (a reading not chosen), and *hintergrund*, a proposal from outside the text, which never carries. The contract is checked by the stage *anmerkung* of the normalise run, which counts violations (*n_vertragsfehler*). In the run reported below it counted none.

3 Stage I: normalise

3.1 The probe is a sentence of its own

The lab reads the probe in the same way as a sentence of the corpus. The text is parsed by Stanza (tokenisation, multi-word tokens, part of speech, lemma, dependencies, named entities), cut into the clauses of its dependency tree, and inserted into the sidecar as a one-sentence letter, *S:probe#0.0* of the letter *probe*. It borrows nothing from any letter of the corpus. If the user supplies a sender, a day or a letter, these enter as context; otherwise the letter record of the probe lacks them, and the normalise stage reports exactly that (*fehlt: ort, tag, absender, empfaenger*). The same code that annotates the corpus annotates the probe, and this is what makes later steps comparable: the probe's keys and the corpus sentences' keys come from one process.

3.2 The nine stages

The normalise stage is a Partitur, a goal tree of nine stages with verification rules, run by the class *Lauf*. Its stages hand a small serialisable summary to each other and to the log.

OPERATION Normalise one probe and the sentences of its sample

Q What does each expression of the probe mean, on what grounds, and how firmly?

1	kontext	loads the letter heads and forms the series of letters of each exchange
2	stanza_lesen	reads phrase candidates from the analysis of the sentence: propositions, predicates, noun phrases, entity spans, pronouns, time words and place deixis
3	ausdruckstyp	applies the ordered type theory: the first class whose factors hold decides, else the phrase stays open
4	aufloesen	applies the resolver of the class and records value, candidates, grounds, assumptions and alternatives
5	kritische_fragen	asks the defeater questions; a background proposal may fill an open phrase, marked as ungrounded
6	urteil	computes the grade from the reasons; no carrying reason, no value
7	anmerkung	builds the footnotes and checks their contract: a label, a rule, an addressed ground for each fixed value
8	schluessel	writes the justified keys of the footnotes to the attribute table
9	monitor	reports counts per stage, the most frequent reasons of open footnotes, and times

→ a footnote with reasons for each expression, justified keys for the filter, and a new version of the normal form

Context. A phrase that the sentence does not decide is looked up in the layers of context, nearest first: the sentence, the paragraph, the letter, the series of letters, the register and usage of the names, and what a curator or an inquiry has added. The first layer that settles it wins. A series is formed from the letters between the same two correspondents, in both directions, or else from the letters of the same folder, and a letter reads a window of ten letters back and three ahead. A reading taken from the nearest letter has the grade `mittel`, and it loses a step for each four letters of distance. A later letter may explain an earlier one, but the reading is then marked as a looking back (`rueckblick`) and capped at `mittel`. A second pass runs over the letters that stayed open, now with the readings of their neighbours.

Type theory. What kind of expression a phrase is, is decided by a type theory in a declarative file. A *factor* is a fact of the syntactic analysis written as a boolean over a row of the phrase table, for example `PRON3` (a third person personal or possessive pronoun), `NER_ORT` (a place entity), `NP_MIT_NAME` (a noun phrase that holds an entity span) and `EINHEIT_KOPF` (a head noun that names an organisational unit: office, branch, laboratory). A *class* holds if one of its conditions, a conjunction of factors, holds, and the classes are tried in a fixed order, so that the more specific class, the named unit, is tried before the plain place. A phrase that no class covers is `offen`. Each class carries a rule code of the edition's terminology guide, of the form `R-XN-...`, so that every footnote can name the rule under which it was made.

Two classes of the type theory (typen.yaml, descriptions line-broken)

```

- klasse: einheit_benannt 1
  beschreibung: a unit named by its head noun 2
                    and a name inside the phrase
  regel: R-XN-I02 3
  bedingungen:
    - {faktoren: [NP_MIT_NAME, EINHEIT_KOPF], quelle: entwurf} 4
- klasse: anapher
  beschreibung: a 3rd person pronoun taking its
                    referent from the text before it
  regel: R-XN-R01
  bedingungen:
    - {faktoren: [PRON3], quelle: entwurf}

```

- 1 Classes are ordered; the first whose conditions hold decides.
- 2 Every class has a plain description
- 3 and a rule code of the terminology guide.
- 4 A condition is a conjunction of factors over the syntactic analysis; the source is marked as a draft of the author, not measured.

Guards. After resolution each footnote is asked a list of critical questions, the *guards*. A guard is a defeater of the footnote's grade, and when it fires its reason is added to the footnote so that the reader sees the objection and what it did. A guard has one of four effects: it lowers the grade by a step (*senkt*), leaves the footnote without a value (*luecke*), withdraws the reading (*stopp*), or is only told (*hinweis*). Examples of the questions are whether two letters contain the sentence, which makes the context of its deixis ambiguous, or whether a date lies more than three years from the day of writing. Resolvers record their own defeaters too: the two gaps of the probe, an anaphor with no noun of the right kind before it and an organisational unit whose owner neither the phrase nor the letter nor the exchange gives, are such reasons, with the effect *luecke*.

Background and inquiry. A phrase that stays open may receive a *background proposal*, from a model (a switch of the context file, off by default) or from a curator, and the proposal has a status of its own, *hintergrund*: it is recorded as a reason of the kind that never carries, so it gives no value, no grade, and no key. A rule may flag a background reading for an *inquiry*, which has four stages: state the reading as a claim with the words that would show it; look in the sources, near to far (the series, the collection, the register, and further sources when a tool for them is given); decide for each hit whether it supports, contradicts or says nothing; conclude. The outcome is *gegrundet* if there is evidence for the claim, none against, and the name is a register individual; *widerlegt* if there is evidence against and none for it; otherwise *unbelegt*. A grounded or refuted outcome is written as an addition, with a *beleg* reason in the first case and a refuting note in the second, and the letter is run again as a new version of kind *erkundung*; *unbelegt* changes nothing but the record. A model never grounds itself.

3.3 The grade

The grade is a five-step scale, *offen*, *schwach*, *mittel*, *stark*, *gewiss*, and it is computed, never set. For a footnote with a value it is the weakest of the grades of its carrying reasons and of its assumptions, lowered by one step for each guard with the effect *senkt*. A footnote with no graded carrying reason, with a *luecke* or *stopp*, or with no value is *offen*; if only a background proposal fills it, its status is *hintergrund*. A footnote is as strong as the weakest thing it rests on. The grades are rule-based: they count the weak supports a reading rests on, they are not a frequency of correctness, and a measure of that belongs to an evaluation of the stage.

3.4 The entry case

The lab ships with one entry case, and this paper uses it throughout. The probe is

The St. Louis office had not been notified in advance of his temporary duty assignment in St. Louis.

and the sample is 200 sentences of the Kármán Nachlass sidecar: four entry sentences, chosen by the author because they mention St. Louis, and 196 further sentences of at least six tokens, drawn by a fixed hash seed. The normalise stage runs over all the sentences of the letters that contain a sample sentence, and over the probe: 85 letters with 1,789 sentences. It produced the figures of Table 3.

Table 3 The normalise run on the slice of the Kármán Nachlass.

Quantity	Value
letters in the run, of which in a series	85, 84
series formed	18
sentences	1,789
phrase candidates	5,784
footnotes	5,500
fixed (fest) / open	4,181 / 1,319
grades stark / mittel / schwach / offen	1,853 / 1,671 / 657 / 1,319
reasons recorded	20,681
guard effects luecke / senkt / hinweis / stopp	1,242 / 540 / 788 / 10
contract violations	0
keys written, channel ausdruck / brief_erbe	2,934 / 5,913
time of the stages, total	5.3 s

The slice shows the problem that the stage solves. In the 85 letters, the letter record lacks the place of writing in 28, the day in 16, the sender in 16 and the recipient in 17. The most frequent reasons that leave a footnote open are anaphors with several nouns of the right kind before them and nothing to decide between them (74 cases for “it” and 15 for “It”), places of writing that the letter record does not give (28), and names for which the register has no individual (for example 18 times for the form “theodore”).

The probe itself receives five footnotes.

Table 4 The five footnotes of the probe.

Phrase	Class	Rule	Status, grade	What the footnote says
“St. Louis” (t2–3)	ort	R-XN-O02	fixed, stark	register individual O000049; nested in the unit, so it says where the office is and is not the referent of the phrase
“his” (t12)	anapher	R-XN-P05	open	no noun of the right kind in the one sentence up to the pronoun (guard: luecke)
“St. Louis” (t17–18)	ort	R-XN-O02	fixed, stark	the same individual O000049, as location of the assignment
“The St. Louis office” (t1–4)	einheit_benannt	R-XN-I02	open	an organisational unit located in St. Louis; neither phrase, letter nor exchange says whose office it is (guard: luecke)
“had not been notified” (t5–8)	praedikat	R-XN-T07	fixed, mittel	past perfect, passive, negative, no modal, assertion; $E < R \leq S$

The footnote of the second “St. Louis” and that of the first have the same key, and the filter will later count them once. The footnote of the predicate lists the frame `praedikat=notify; thema=The St. Louis office; ort=advance`, in which the temporal adverbial “in advance” fills the place slot. This is a defect of the frame reader, not of a rule file. The learning loop will meet it in Section 6 and will refuse, correctly, to repair it by editing a rule.

The probe’s normal form is the XML below (line breaks added, and the attribute `ref` of the marks `e3` to `e5` omitted). Its two open footnotes appear in the `normal` view as explicit gaps, never as guesses.

```
<p><expr typ="einheit_benannt" status="offen" grad="offen" regel="R-XN-I02"
  normal="the office, in St. Louis, Missouri, United States (0000049),
    of an organisation not named"
  ref="S:probe#0.0@1-4:einheit_benannt" id="e1">The
  <expr typ="ort" status="fest" grad="stark" ind="0000049" kind="place"
    rolle="ort_von" regel="R-XN-O02"
    normal="St. Louis, Missouri, United States (0000049)"
    ref="S:probe#0.0@2-3:ort" id="e2">St. Louis</expr> office</expr>
  <expr typ="praedikat" status="fest" grad="mittel" regel="R-XN-T07"
    id="e3">had not been notified</expr> in advance of
  <expr typ="anapher" status="offen" grad="offen" regel="R-XN-P05"
    normal="['his': no antecedent found]" id="e4">his</expr>
  temporary duty assignment in
  <expr typ="ort" status="fest" grad="stark" ind="0000049" kind="place"
    rolle="ort" regel="R-XN-O02" id="e5">St. Louis</expr>.</p>
```

The `normal` view of the probe is: *the office, in St. Louis, Missouri, United States (0000049), of an organisation not named had not been notified in advance of ['his': no antecedent found] temporary duty assignment in St. Louis, Missouri, United States (0000049)*. This is the string the classifier reads in the third stage.

The version store holds the probe as a letter of its own, `probe-3165295f`, and its history shows what the five kinds of change are for (Table 5). Version 2 is a curator’s addition: the owner of the office was named as Curtiss-Wright, which fixed the open footnote of the unit. It was then withdrawn in version 3, and the run reported in this paper reads version 6, in which no addition is in force and the owner is open again. The version store thus records that a curator can close this gap, and that the gap was open again in the run reported here.

Table 5 The history of the probe in the version store; the kind is the field `art`.

No.	Kind	Reason recorded
1	<code>lauf</code>	first run
2	<code>kuration</code>	curator names the owner (Curtiss-Wright, institution I000001)
3	<code>kuration</code>	the curator’s additions changed (none remain)
4	<code>lauf</code>	re-run with the nested normal form
5	<code>kontextaenderung</code>	re-run after review fixes, with a smaller window (5 back, 2 ahead)
6	<code>kontextaenderung</code>	the context changed (default window 10 back, 3 ahead)

METHOD What the normalise stage does not claim

The grades and the type theory are drafts of the author, marked `quelle: entwurf` in the rule files. The counts of Table 3 are counts of the stage, not measures of correctness; no human has checked the footnotes of the slice. The contract that no value stands without a ground is verified, and the correctness of the grounds is not.

4 Stage II: filter

4.1 Why a filter, and why it comes second

The classifier of the third stage is the expensive and fallible part of the lab. A relation search over every sentence of an edition would call it once per sentence; the companion paper counts 663,972 sentences in the Kármán Nachlass. The filter reduces the candidates by a criterion that is cheap, exact and decided before any model reads a sentence: the sentence must share a *justified key* with the probe. The filter reads the keys of the normalise stage and does nothing else; it normalises nothing, and a filter variant names the normalise variant whose result it reads.

This division is also a division of kinds of irrelevance. A sentence that shares no justified person, place or institution with the probe is *referentially* irrelevant: it is not about anything the probe is about, and nobody has to judge it. A sentence that shares the place and says nothing about the probe’s claim is *logically* irrelevant, and that is a matter for the third stage (Section 5). The first kind is removed by a join; the second kind needs a reader.

4.2 From footnotes to anchors

A footnote of the probe becomes an *anchor* if its kind is among those that the filter file lists (`place`, `person`, `institution`), its status is fixed, and its grade is at least the minimum (`mittel`). The function that builds the anchors also does four things that are visible in the declarative file.

- **Open footnotes can contribute their candidates**, weakened to `schwach` and joined by OR within the footnote (`offene_kandidaten`), but do not by default.
- **Nested footnotes can be switched**: a place that only says where a unit is (`ort_von`), and an owner named inside a unit (`besitzer_von`), anchor like any other place or owner (`anker`) or not at all (`nein`), depending on whether the filter should find sentences about the unit or about the place. In the default file both anchor.

- **Equal keys are one anchor.** The two footnotes of “St. Louis” in the probe, one nested in the unit and one locative, have the same key and give one anchor.
- **Letter metadata of the probe** (its sender and its day of writing) become anchors over the channel `brief_erbe` only if the probe has a letter context. Nothing is borrowed.

The anchors are joined by AND (`alle`) or by OR (`eines`); the entry case has one anchor, so the choice is not exercised there.

4.3 One bound query

The filter is one SQL statement of the following shape, instantiated for each anchor, and the instantiation is shown with the filter result so that a reader sees what was asked.

```
SELECT s.obj_id, s.text
FROM sr_sentence s
WHERE s.obj_id IN (SELECT unnest($1))           -- the 200 sample sentences
  -- anchor 1: 'St. Louis' -> St. Louis, Missouri, United States (0000049) [stark,
R-XN-002]
  AND EXISTS (SELECT 1 FROM sr_attr a
              WHERE a.obj_id = s.obj_id AND a.kind = 'place'
                AND a.key_id IN (12158054552581156312)
                AND a.kanal IN ('ausdruck', 'ner_linked')
                AND (a.cert IS NULL
                    OR list_position(['offen', 'schwach', 'mittel', 'stark', 'gewiss'],
a.cert)
                    >= list_position(['offen', 'schwach', 'mittel', 'stark', 'gewiss'],
'schwach'))))
ORDER BY s.obj_id
```

A corpus sentence matches on a key written by the normalise stage (`ausdruck`) or by the register’s own linking of entity mentions (`ner_linked`), and the grade of the key in the corpus sentence must reach a minimum. If the probe has no anchor, the query gets `AND FALSE` with the comment that the probe has no justified key of the filter’s kinds: such a probe keeps nothing, and the lab says so rather than keeping everything.

4.4 The entry case

The filter reduced the sample from 200 to 4 sentences in 0.09 s. The four survivors are the entry sentences of the lab, and they are shown with their texts in Table 6, because the third stage works on exactly these.

Table 6 The four survivors of the filter.

Sentence	Text (original)	Gold reading
#18.6	I believe you recommended that he be hired for the Design Board in St. Louis.	independent
#18.10	He went to St. Louis and returned to Pasadena.	independent
#18.11	When he returned he called me and said he had been to St. Louis and no one there expected him or knew what to do with him.	abductive
#18.14	Hovatter I finally found out that he apparently did n't make a good impression in St. Louis among Smith, Shoults and the rest.	independent

The error matrix against the gold file counts true positives 4, false positives 0, false negatives 0 and true negatives 196. It must be read with care. The gold file records, in its own words, “the author’s reading of the entry case, not human gold”; the four sentences that must stay are the four entry sentences that were put into the sample by construction, and every other sentence of the sample counts as one that must go. The matrix therefore shows that the filter retains the sentences it was designed around, and that 196 sentences drawn by hash share no justified St. Louis key with the probe. It does not show a recall on sentences that nobody chose beforehand, and the sample contains no sentence that mentions St. Louis in a form the register does not know.

The probe’s letter metadata (sender and day) could narrow the result further, as anchors over `brief_erbe`. In the run of this paper the probe has no letter context, so none of these anchors is active.

5 Stage III: classify, with the relation catalogue of famulus-logic

5.1 The four cells

The relation of a stored sentence *S* to the probe *P* is read off four *cells*, the joint truth-value situations TT, TF, FT and FF of *S* and *P*. A label of the catalogue names the cells that are *impossible*, and no two strict labels name the same set. The classifier prompt of the lab asks for both: the truth value of each cell, and the label, so that a label that the cells do not support can be seen.

Table 7 The seven strict labels.

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

The cells are judged under logical necessity: a consequence counts only if it holds in every situation in which S is true, with no hidden assumption, no “usually” and no “typically”. Thematic similarity, shared words and shared names are never by themselves a ground of a relation. The needle suite of the companion paper turns this rule into a test: it contains close-but-negative pairs, *traps*, built to look like a relation they do not have, and the *trap rate* is the share of them that a classifier calls related.

The user of the lab asks three kinds of question, and the catalogue answers each by a different region of the table.

Entailment. The probe “The report was written in 1938” is entailed by “written in March 1938 in Pasadena”, and the three labels `entails`, `entailed_by` and `equivalent` carry the direction. A stored sentence that entails the probe is a *support* of it; one that the probe entails is a *consequence* the scholar may look for in the edition.

Contradiction and opposition. The catalogue distinguishes *contradictory*, where exactly one of the two holds, from *contrary*, where both may fail, and from *subcontrary*. The distinction is not scholastic. “Written in 1939” is contrary to the probe “written in 1938”, not contradictory to it: the report may have been written in 1937. A relation search for contradictions that cannot tell the two cannot tell a refutation from a competing statement, and, as the companion paper measured, a cross-encoder trained on three-way inference does not tell them: the inability to separate `contrary` from `contradictory` accounts for 53 of its errors on the catalog slice of the needle suite.

Independence, which the lab calls irrelevance. The label `independent` means that no cell is impossible: S gives no information about the truth of P and P none about S. An honest classifier must be able to say it. The survivors of a filter are the sentences that are *referentially* related to the probe, and many of them are logically independent of it: in the entry case three of four. A classifier that calls a sentence related because it mentions St. Louis has made the mistake that the traps of the needle suite are built to catch.

■ 5.2 The weak test and the strength axis

The strict labels leave no room for a default. A scholar of letters needs more, because the editorial reading of one sentence by another is often defeasible, and so the classifier has a weak tier beside the seven strict labels, and the catalogue has the label `abductive`, which is not truth-functional. The classifier prompt has two steps. Step 1 gives the strict label. Step 2, run only if the strict label is `independent` or `subcontrary`, asks whether S gives defeasible support to P: either that P follows from S by a default or an implicature (`weak_relation: entails`), or that P is the best explanation of S under a stated hidden assumption that a defeater could cancel (`weak_relation: abductive`). Topical overlap, correlation, post hoc reasoning, an open choice among equally good explanations and an overreaching P are not support, and the prompt says so. The answer carries a *strength* on an axis that runs necessary, presupposed, hedged, implicated, defeasible, and the axis is independent of the label: one pair may be strictly `independent` and weakly `entails` at the strength defeasible.

CONCEPT Strict and weak are two answers, not one

A label that is decided by the four cells can be certified. A label that rests on a default, an implicature or an explanation can be reported, and it must carry its strength and the hidden assumption on which it rests, so that it is never read in the voice of a necessary consequence. The lab reports the strict label, the weak label and the strength separately for each survivor, and combines them only by a rule that is written in a declarative file.

■ 5.3 What the classifier reads

The input of the classifier is not the raw sentence. The lab’s setting `textform: normal` makes the classifier read the `normal` view of each survivor and of the probe, the one in which the expressions of the first stage are explicit. Three consequences follow, and the entry case shows each of them.

First, indexicals are resolved, with their individuals. In #18.10 “He went to St. Louis and returned to Pasadena” the classifier reads *Hovatter, Maj. went to St. Louis, Missouri, United States (O000049) and returned to Pasadena, California, United States (O000007)*. The probe’s St. Louis is the same individual, and the classifier does not have to establish it.

Second, a gap stays a gap. In #18.6 the pronoun is rendered as [*‘he’: 14 possible antecedents, not decided*], and in #18.11 the “there” is rendered as [*the exchange names 18 different places and nothing decides between them*]. A classifier that sees this cannot silently make “there” be St. Louis, which is the step that a fluent model would make, and the unresolvable parts of the annotation become part of the question. This is the point at which the expression layer protects the relation layer.

Third, whether the normal form helps a cheap classifier has been measured, and the result is modest; we repeat it as it is. On the needle suite the renormalised input of the cross-encoder gave no gain (strict accuracy 0.471 against 0.475, within noise), because the needle items are synthetic sentences without a parse, so the clause mode could not be run on the suite; and on 150 real multi-clause sentences of the slice, where there is no human reference, the clause mode changed 18 per cent of the labels against the raw mode, which says that it reads differently and not that it reads better. The mode that cuts a sentence into the clauses of its dependency tree composes the sentence label monotonically: S entails P if some clause does, P entails S only if P entails every clause, and S contradicts P if some clause does. A *relevance gate* admits a contradiction only from a clause that shares a content word with the probe. With the gate the clause mode gave 2 contradictions on the 150 sentence pairs, without it 58. Whether the 56 withheld contradictions were wrong has not been checked by hand. In the entry case the cross-encoder reads the raw form (`nli_form: raw`), and the classifier’s `textform: normal` concerns the language models only.

5.4 The cascade

The classifier of the lab is a cascade of four tiers. Each tier is the cheapest that can be certain, and each pair records the tier that decided it. The cascade is the one of the companion paper.

Table 8 The four tiers of the classification cascade.

Tier	Component	Decides when
T1	the Famulus Normal Form, compiled by a model	it returns a decisive strict label (anything but independent or unknown), or a weak label
T2	bidirectional cross-encoder (mDeBERTa-v3)	T1 said independent or unknown, the cross-encoder sees no signal in either direction (entail and contradict both below the floor 0.15) and the two sentences share no content concept: the pair is independent
T3	a fast model with the two-step prompt	otherwise: it gives the strict and the weak answer
T4	a strong model with the same prompt	T3 returned unknown, or T3 and the cross-encoder disagree on whether any strict relation holds

The routing is a property of the cascade that the companion benchmark measured. On a subset of 160 pairs of the needle suite with `claude-haiku-5-5` as the fast tier, 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 strict accuracy was 86.3 per cent, the same as the fast judge alone, and the trap rate was 3.8 per cent. The cascade with that fast tier spares a quarter of the judge calls and gains no accuracy. With `gemma-3.5-flash-lite`

as the fast tier the accuracy was 69.6 per cent and the trap rate 23.8 per cent: the cascade is bounded by its fast judge. The cross-encoder tier closes almost nothing, because content words are nearly always shared between two sentences of one edition.

In the lab the cascade is configured by a declarative file, `klassifikation.yaml`, which names the models of the tiers, the floor of the cross-encoder, and the form the cross-encoder reads (`raw`, `norm` or `clause`). The lab's model field overrides all models of the cascade at once. In the run reported here every tier used `claude-haiku-5-5`, so that the fast and strong tiers differ only in their role in the cascade and not in the model.

5.5 The entry case

For each of the four survivors the cascade answered as in Table 9 (variant A of the classify stage).

Table 9 Classification of the survivors by the unmodified prompt.

Sentence	Tier	Strict label	Weak label	Strength	Gold
#18.6	strong	independent	none	none	independent
#18.10	strong	independent	none	none	independent
#18.11	strong	independent	none	none	abductive
#18.14	fast	independent	none	none	independent

Three pairs were settled at the strong tier and one at the fast tier. The routing rule of the cascade implies a reading of this: for #18.6, #18.10 and #18.11 the fast judge said `independent` and the cross-encoder said that some strict relation held, so the strong judge adjudicated, and it sided with the fast judge (the record stores the fast label, not the cross-encoder's probabilities, for these pairs). For #18.14 the probabilities of the cross-encoder were (entail, contradict, neutral) = (0.057, 0.019, 0.924) forward and (0.040, 0.023, 0.937) backward: it saw no signal, but the two sentences share content words, so T2 could not close the pair, and the fast judge's `independent` agreed with the cross-encoder and stood without escalation. The sentence is an instructive independent case: it says that Hovatter made a poor impression in St. Louis, it shares the place and the person with the probe, and nothing it says bears on whether an office was notified.

The gold has one abductive pair. The unmodified prompt misses it: the abductive count is TP 0, FP 0, FN 1. The sentence #18.11 is the one in which the traveller "called me and said he had been to St. Louis and no one there expected him or knew what to do with him". P would explain S: had the office not been notified, nobody there would have expected him or known what to do with him. The explanation is defeasible, with the rival that a notice was sent but did not arrive, and the weak test is built for exactly this. The test of how far a prompt can move that pair is in Section 6.

5.6 Inference over more than one sentence

The relations above are pairwise. The lab adds two inferences whose premises lie in more than one sentence.

Cross-sentence abduction. The abductive rules of the Famulus Normal Form read one sentence. A letter says "The instrument has a graduated arc" in one place and "It has a sighting tube" in another, and a rule that needs both finds each in a different sentence. The module `fnf/abduktion.py` of `famulus-proof` finds *siblings* by keys, not by words: two sentences of one carrier (a letter) are siblings if they share a person who is meant and not merely named in an epithet, a place, an institution, a concept, or a date whose interval overlaps. Siblings form connected groups, the strict literals of a group (base and presupposed) are pooled, and the abductive rules run on the pool. The result is one of three kinds. A conclusion that needs premises of two or more sentences is `quer`, in the support catalogue `linked`: the premises carry the conclusion only together. The same conclusion abducted from two sentences on their own is `konvergent`: each gives a reason and neither is needed. A conclusion that a sentence would abduce on its own but that a sibling's stated rival withdraws is `blockiert`, with the rival named. A conclusion is never strict: its grade is A, it carries its hidden premises, still to be granted, and its rivals, those stated in the carrier and those left open. In the lab

the probe and its survivors form one group of siblings, and a survivor that is linked to the probe by a `quer` or `konvergent` conclusion is raised from `independent` or `subcontrary` to `abductive` by a declared rule (`abduktion.anheben` in `klassifikation.yaml`).

In the entry case this inference produced no conclusion. The pooled strict literals of the survivors and the probe were 3 (`#18.6`), 2 (`#18.10`), 7 (`#18.11`), 3 (`#18.14`) and 1 (the probe), and the hand-written abductive rules did not fire on them. The same absence appears in the pilot over the sidecar: 154 groups of siblings with 982 sentences (the largest group has 37), 4,385 sibling pairs, and no conclusion. The cross-sentence derivation is covered by synthetic tests; on real data of this corpus it has not yet produced a conclusion, and we do not know whether the rules or the sample are the cause.

The three-sentence test. A probe is also supported when two sentences support it together although neither does alone. The lab takes each pair of survivors, A and B, with the probe as the conclusion C, and asks a judge which truth-value cells of (A, B, C) are impossible, from which a support label follows (for example `linked`, `convergent` or `no_support`) under a standard of warrant. Where the support is lacking, the judge must name the *missing premise*. All six pairs of the four survivors came back `no_support`, and the missing premises are informative as annotations. For (`#18.10`, `#18.11`) the judge wrote that a warrant is needed that links expectation to notification, “if the St. Louis office had been notified in advance of Hovatter’s temporary duty assignment, then Hovatter’s arrival in St. Louis would have been expected there”, and that no such warrant is stated in the source. For (`#18.10`, `#18.14`) it asked for a bridging premise that identifies the St. Louis trip as the temporary duty assignment in question, together with the claim that the office would have been notified in advance of any such assignment. The judge is `claude-haiku-5-5` and the six calls took 3.3 to 4.9 s each. This is the critical-apparatus function of the third stage in its purest form: it does not say that the premises are false, it says which sentence the edition would have to contain for the inference to be warranted.

6 Learning the rules: gated A/B variants

6.1 One loop for three modes

The three stages differ in what they read and in the files of rules that govern them, and each has its own loop. A *variant* is a directory under `runs/labor/varianten/<mode>/<name>/`. Variant A is the package’s own files. A learning variant B<n> holds the files that its instruction changed, the Partitur of its mode with a record of the instruction, the edits and the model that made them, and its result. A result records the variants of the upstream stages that it read, so that a filter result names its normalisation and a classification result names its filter.

Table 10 Per mode, the files that learning may edit, and the gate.

Mode	Files a variant may change	Gate
normalise	word lists, type theory, guards, context layers, wording of foot-notes	<code>tests/test_ausdruck.py</code> and <code>tests/test_normalform.py</code> with the variant’s files
filter	the filter file (kinds, minimum grade, open candidates, channels, nesting, join, letter metadata)	<code>tests/test_probe_filter.py</code> with the variant’s files
classify	the cascade file, the two classifier prompts, the abductive rules	the four placeholders in both prompts, a valid cascade file, loading of the catalogue of <code>famulus-logic</code> , <code>tests/test_abduktion.py</code> with the variant’s rules

The learner is a language model (`claude-sonnet-5-5`) that receives the instruction, a state taken from the basis variant’s result, and the text of the files it may change. It must answer in JSON with a list of edits, each of an operation (`set` , `add` , `remove` , or, for a prompt, `replace` of a text that occurs exactly once), a dotted path, a value and a reason, and with two statements that the lab requires: what it *expects* to change in the result and what the edit *cannot do* (`grenzen`). It may not change code, invent registry keys, or state facts about the corpus; if the instruction needs a code change it must say so in `grenzen` and propose nothing for that part. Edits are applied to a copy, the gate is run with the variant’s files in place, and only then is the variant run with the same upstream choices as its basis. A variant is adopted by copying its files over the package’s, with the old files archived and the basis re-run.

One detail makes a comparison of A and B meaningful: the pair judgements are cached by classifier, options, the hash of the prompt files, the probe and the text of the sentence, so that the same input gets the same label in A and B and a difference comes from the part that was changed and not from a model’s variation between runs. A changed prompt changes the hash, so the pairs are judged anew exactly where the edit applies.

6.2 What the recorded variants show

The entry case has been through the loop several times, and the records are instructive less as improvements than as a demonstration of what the declarative boundary does.

Normalise B2. The instruction was to let the filter also use institution footnotes of the probe, and to make the head noun “office” count as an institution word. The learner noted that the filter file already lists `institution` , so an edit would change nothing there, and for the second part it proposed no edit, for three reasons: phrases come from the syntactic sources, and no word list can create a phrase row for the head noun; an institution footnote needs a registry key and “the St. Louis office” has none that may be invented; and a list and a factor added anyway would at best stay unused and at worst give false institution readings to every “office”. It said what the change would be: a code change that makes noun heads into phrase rows linked to registry institutions. The variant has zero edits and passes its 14 tests.

Normalise B3. The instruction was to make the footnote of the open “his” say plainly that the probe is a single sentence, so that no antecedent can exist in it. The learner changed one template, `w.kein_antecedens` , and the footnote now reads *The probe is a single sentence with no sentence before it, so no antecedent can exist in it: no noun of the right kind (person) can occur before the pronoun ‘his’ (1 sentence(s) searched, the sentence itself included)*. The grade and status stayed open. The learner noted in `grenzen` that the template is shared by all open pronouns, so that in a longer text where a noun is simply missing the new wording would be false; a conditional wording needs a code change or a separate reason key. The gate (39 tests) passed. The edit is correct for the probe and wrong in general, and the record says so.

Filter B1 and B2. Both set `offene_kandidaten` to true. The instruction of B1 had two parts: that “in advance” is a temporal adverbial and must not fill the place slot of the frame, and that the filter should use the candidates of the open “his” if there are any. The instruction of B2 had only the second part. The filter result is the same as A in both: 200 to 4, TP 4, FP 0, FN 0, TN 196, and the same single anchor. The learner predicted this: the open footnote of “his” lists no candidates, because no person noun occurs before it, so no anchor can arise from it. For the first part of the instruction B1 proposed nothing: the frame slots are filled from the dependency analysis in code, and a guard cannot repair them, because a guard changes the grade of a whole footnote. B2 added a warning of its own: with the join `alle` an additional anchor narrows the result and cannot widen it.

Classify B1. The instruction was that a survivor that reports a person going to or being in the place of the probe, where the probe says that an office was not notified of the person’s trip, is an abductive explanation, and that the weak test must name the strength. The learner replaced one line of the weak prompt by a paragraph that defines the visit case, requires the hidden assumption (the visit was made without prior notice to the office), limits the rule to sentences about the visit itself or its being known, excludes sentences that only mention the place, and requires a strength for any weak label. The gate (placeholders, cascade file, catalogue, and 8 abduction tests) passed.

Table 11 Variant B1 of the classify stage against variant A.

Sentence	A: joint verdict	B1: strict, weak, strength	B1: joint verdict	Gold
#18.6	independent	independent, none, none	independent	independent
#18.10	independent	independent, abductive, defeasible	abductive	independent
#18.11	independent	independent, abductive, defeasible	abductive	abductive
#18.14	independent	independent, none, none	independent	independent

With B1 the abductive counts are TP 1, FP 1, FN 0, against TP 0, FP 0, FN 1 with A. The target sentence is found, and #18.10, “He went to St. Louis and returned to Pasadena”, is also called abductive, though its gold is independent: it reports a visit and says nothing about anyone’s expectation. The learner had warned of this in its `grenzen`: it limited the rule to sentences about the visit and its being known, “but the prompt cannot guarantee that”. Both pairs were settled at the strong tier, and the classification run of B1 cost USD 0.029 and took 22.8 s. The result teaches what an A/B variant is for. The single prompt edit moves a recall of 0 of 1 to 1 of 1 at a price of one false positive, and the two further devices of the lab, the cross-sentence abduction and the three-sentence test, did not contribute on this case: in B1 as in A the abduction gave no conclusion and all six pairs came back `no_support`. The case is also tiny. One probe, four survivors and one abductive gold pair are an existence proof of the mechanism and not a measurement of its effect.

7 Critical annotation as a service of the reference storage

7.1 The annotation record

Put together, the three stages produce for a probe and an item an annotation that has the following parts. The table is a design statement: the parts exist in the lab’s files, and the single record that joins them, addressed by an `fr`: address of the item, does not.

Table 12 The parts of an annotation of an item against a probe.

Part	Content	Produced by	Kept in
Item address	the sentence address of the sidecar (Section 2), to be mapped to an <code>fr:</code> address	the sidecar	sidecar
Expression foot- notes	quote, rule, label, value, grade, status, reasons, assumptions, defeaters, alternatives	normalise	<code>sr_expr</code> , <code>sr_expr_grund</code> , XML normal form
Normal form	the item's text with marks and back block, versioned, with the three views	normalise	version store
Keys	justified numeric keys by channel	normalise	<code>sr_attr</code>
Anchor trace	which footnote of the probe made which anchor, and the bound query	filter	result file
Relation	strict label, weak label, strength, tier, certificate or cells	classify	result file
Inference over more sentences	cross-sentence conclusion with premises, assumptions, rivals; support cells with missing premises	classify	result file
Provenance of the annotation	variant, rule file hashes, context hash, model, prompt hash, cost	the lab	Partitur, version store

An answer built from such records has the property that the reference storage was designed to give citation: a reader who doubts a verdict can follow it to the sentence, to the footnote that fixed the reference, to the reason that carried the footnote, to the rule file and its hash, and to the model and prompt that judged the pair. And for each of these steps the record says what is missing.

7.2 The invariants as a recipe

The conceptual content of the lab may be stated as a recipe that does not depend on the particular rules.

METHOD A recipe for critical annotation of the items of a reference storage

1. **One process.** Treat the claim and the stored sentences by the same process, so that their annotations are comparable.
2. **Expressions first.** Annotate each expression as a quoted span with a rule, carrying reasons, assumptions, defeaters and alternatives, and compute the grade from them; a span with no carrying reason has no value and is told to be open.
3. **Outside the item.** Keep the annotation as a version of a derived document that can be removed to give the original; never overwrite, always give a reason.
4. **Background is not evidence.** Whatever is not in the text, the model's knowledge and a curator's hunch, is a proposal of its own status that never carries a value; only an inquiry that finds evidence may ground it.
5. **Keys before readers.** Reduce the candidates by justified keys before any reader sees a sentence, and report the anchors and the query.
6. **Three regimes, two strengths.** Judge the relation of the survivors by the four cells, keeping entailment, opposition and independence apart; keep strict and weak apart, with a strength and the hidden assumption of every weak verdict.
7. **The normal form as input.** Give the classifier the normal form, so that the gaps of the first step are gaps of the question.
8. **Several sentences, same discipline.** Name the premises, list the assumptions, name the rivals, and never call the conclusion strict.

9. **Rules are data.** Edit the data by instruction, gate the edit, and compare it with its basis on cached judgements.

7.3 Limits and next steps

The paper's claims are bounded by what has been measured, and the bounds are narrow.

The empirical record of the lab is one probe, a sample of 200 sentences of the Kármán Nachlass sidecar, four survivors and one abductive gold pair, and the gold is the author's reading of the entry case, not human gold. The counts of the normalise stage are counts and not accuracies, its grades and type theory are drafts, and no human has checked its footnotes. The filter result is a designed case. The classification measures of the cascade (the routing shares, the accuracies, the trap rates) are those of the companion benchmark on its needle suite of 703 pairs, whose gold is generated and partly judge-agreed, and not on this slice. The hand-written abductive rules did not fire on the sample, nor in the pilot over the sidecar, which is an open question: either the sample is too small, or the rules do not fire in this corpus. The lab's annotations are not yet stored in the reference storage under `fr:` addresses.

Three pieces of work follow from this. The first is to fix the mapping from the sidecar's sentence address to the `fr:` address and to store the annotation record of Table 12 in the DuckLake at a named snapshot, so that the annotation is an item of the storage and not a neighbour of it. The second is a human review of a sample of footnotes and of verdicts, which is the only way to turn the lab's counts into accuracies; the review queue of the benchmark exists and has not been signed. The third is the repair of what the learning loop could name but not change: the frame reader that puts "in advance" into the place slot, and the phrase rows for head nouns that would let "the St. Louis office" become an institution anchor. These are code changes, and the loop's habit of saying so in `grenzen` is the clearest evidence that the boundary between data and code was drawn where it matters.

8 Conclusion

A reference storage answers where a text is and where it came from. To answer what it says, and how it stands to a claim, the sentence must first become a statement with fixed reference and the claim must be treated in the same way. The Probe Lab does this in three stages. Normalise gives each expression a footnote with grounds, grade, objections and alternatives, under the rule that no value stands without a ground, and keeps the result as a reversible, versioned normal form. Filter reduces the candidates by the justified keys of the probe to a set small enough to be read. Classify assigns each survivor one of the regimes of the four-cell catalogue, entailment, opposition and independence, with a weak test and a strength for what is defeasible, and states, for what needs several sentences, which sentence the edition would have to contain. The stages are governed by declarative files that a gated loop may change and that refuse changes that belong to code. What the lab demonstrates is a mechanism, on one case. What it asks of an edition is modest and exact: that a reading be quoted, that it be grounded, that it be open to objection, and that where nothing grounds it, it say so.