Famulus Harness

A Notebook Beside a Conversation: Instruction Sessions, Imported Models and Checked Changes

Gerd Graßhoff iD
Humboldt-Universität zu Berlin, Deutsches Museum
gerd.grasshoff@hu-berlin.de

2026-11-12

Abstract

A scholarly notebook holds a text, the computations it rests on, and objects drawn from formal sources: an animated figure of an astronomical model, the checked steps of a geometrical argument. A conversational agent can change such a notebook quickly, and that is the difficulty: a change is useful only if it is checked before it is seen, attributed afterwards, and reversible to the byte. This paper describes the Famulus Harness, a desktop application in which a Famulus notebook stands beside a conversation, and the mechanism that makes changes safe: the instruction session. The author selects one or several segments of the notebook (a paragraph, a table, a cell, a figure), says in words what is to change or chooses a preset, and the result is checked and built into the text at once; the session goes on with the next instruction until the author accepts or aborts, and aborting restores the state before. The engine is chosen from a catalogue by the kind of segment. A notebook project declares what it imports: other notebooks, issues of a publication series, and packages of formal texts such as the models of the language astra/1; it can extend an imported package by texts of its own, including new models. We follow one notebook throughout, a working copy of a paper on geometrical reasoning with the proof that the eccentric and the epicyclic form of Ptolemy's solar model are equivalent. On 11 November 2026 a figure of that proof was given a line of sight and a narration in stages by one instruction (28 s), a new model with its superposition, its values and its figure was written in one turn (65 s) after the missing witness had been named, a cell was changed and computed again, and every aborted session left the 13 files of the project identical by SHA-256. A selection can also be annotated, in a database of the notebook, and explained: each kind of explanation is an editable score that names its context, and one kind reconstructs the argument of the selection from the notebook before it and has it checked by the prover the notebook already uses (12 November 2026). What is not built, and the defects found during this validation, are listed as such.

KEYWORDS scholarly notebooks · conversational agents · checked edits · version history · history of astronomy · geometrical reasoning

1 An instruction and what it takes to carry it out

The notebook this paper follows is a working copy of *Geometrical Reasoning in the History of Astronomy* [1], the first issue of the series Famulus Expertise Astra. One of its sections treats what a figure supplies to an argument without saying so. Its example is the proof that the two forms of Ptolemy's solar model, the eccentric and the epicyclic, place the sun at the same point when the epicycle has the eccentricity as its radius and the deferent has the radius of the eccentric [2, 19]. The notebook shows the figure of that proof in motion. The author, reading it, gave this instruction (in German; the notebook is in English):

> Build in a red line of sight from the observer O to the sun S. The animation is to be commented as a living annotation, in stages with a running text under the figure: first the one model shown and described, then the second, then the superposition of both, then the line of sight, then the whole animation through one revolution, to demonstrate the equivalence visually.

To carry this out, several things must hold at once. The figure is not a picture in the notebook. It is drawn from a text in the language astra/1 [3], which belongs to a shared collection of models and must not be

changed for one notebook. The change needs a text of the notebook's own, written in a formal language with its own rules, checked by that language's checker, built, and shown in place of the old figure. The author must be able to look at the result, say "also show the sun at the apogee", look again, and at the end keep everything or get back exactly what stood before. And afterwards the notebook must say who changed what, by which instruction.

The Famulus Harness is the application in which this is done. It serves the third pillar of the lead thesis of the Famulus papers [4], research in a group around shared and addressable notebooks, and it shows the second pillar at work on the scholar's own text: no change is built into a notebook unchecked, and every change states its grounds. The notebook format, its cells and its publication are the subject of the Famulus Notebook book [5]; this paper describes only what the harness adds: the notebook beside a conversation (Section 2), the segment as the thing one works on (Section 3), the instruction session (Section 4), the versions of all writers (Section 5), imported packages and a project's own models (Section 6), cells that compute with them (Section 7), addresses (Section 8), and the agent with its tools, commands and approvals (Section 9). Everything described was run on the working notebook on 11 November 2026; the protocol of that validation lies beside the notebook. Section 11 lists what is not built and what went wrong.

2 The harness and its parts

A *harness* is a program that runs a language model as an agent: it holds the conversation, offers the model tools, asks the user before a tool changes something, and shows the result. The Famulus Harness is such a program for one purpose, work on Famulus notebooks. It is assembled from parts that exist independently (Table 1).

Table 1 The parts of the Famulus Harness as installed on 11 November 2026.

part	what it is	what it does here
the application	a macOS program, *Famulus Harness*, that starts its own instance of DeepSeek Harness (version 0.2.2-rc.1) with a state folder of its own and shows its page in a web view	conversation on the left, notebook on the right; the desktop harness of the same vendor and its sessions are not touched
the plugin, host half	`@famulus/dsh-famulus-notebook` 0.7.0, loaded by the harness	38 tools for the agent (32 for notebooks, 6 for references), 10 commands for the user, the gate that asks before a write, a section of the system prompt, the events that tell the page what to show
the plugin, page half	components placed into the harness page	the notebook pane beside the conversation, the library as a main page, the Famulus name and colours in two modes
the notebook service	a local web service (port 5118) with the notebook viewer	reads, runs and writes notebooks; holds the instruction sessions, the version journal, the imports
famulus-astronomica	the language astra/1 with checker, renderer and animation runtime (package `astra` 0.3.0)	checks and builds every astra text a session writes; answers the cells that ask it
the reference service	resolver of `fr:` addresses (famulus-ref/2, vocabulary 2.6.0)	resolves notebooks, cells and artefacts by address

The division matters for what follows. The agent in the harness never writes a notebook file. Its tools ask the notebook service, and the service is the only writer: it checks, writes, records. The same holds for the reader

who works in the notebook pane without any conversation; the pane is the service's own viewer in a frame. An instruction session can therefore be opened from three places, the panel of a selection, a command typed into the conversation, and a tool the agent chooses, and it is the same session with the same checks.

The agent of the conversation is DeepSeek V4.1 Flash, served by a local Ollama with its reasoning trace switched off; this is the default of all text work on the platform [4]. The engines that carry out instructions on segments are a separate matter (Section 4).

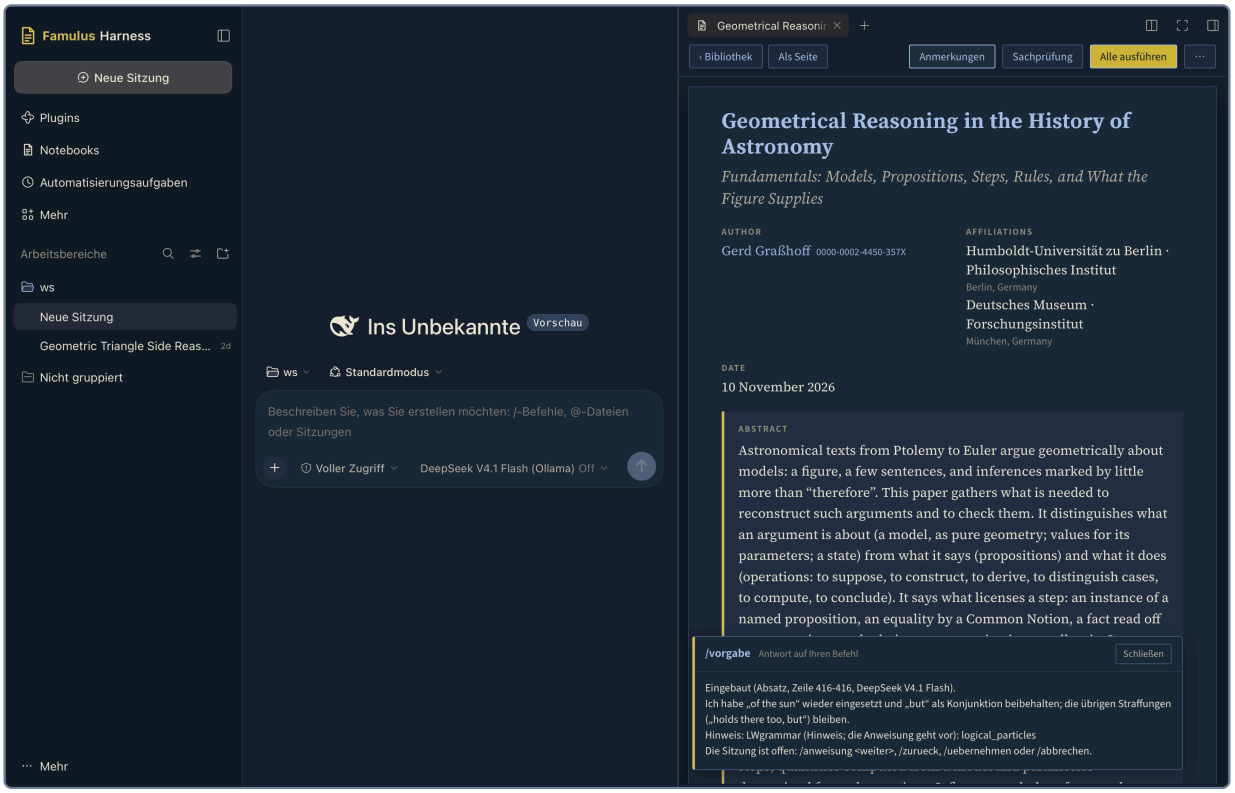


Figure 1 The Famulus Harness with the working notebook beside the conversation (dark mode of the application). The card at the foot of the notebook pane is the answer to a command the user typed, `/vorgabe lektor:straffen "Degenerate states are where figures mislead"`: built in, with the engine, what was changed, and a note of the language check.

3 The segment as the thing one works on

A notebook is read as a paper is read: headings, paragraphs, citations, figures and tables with captions, cross-references; between them cells with their results, and astra segments, which show an object of famulus-astronomica through a line such as `{{< famulus fr:astra/solar-equivalence-equal-radii argument >}}`. Each of these is a *segment*: a span of lines of the notebook's source that the viewer shows as one unit.

Earlier versions of the viewer offered several entry points at a paragraph: a mark for an annotation, a pencil for editing, a bar over a selection, and two buttons in the bar above the text, "Anmerkungen" and "Sachprüfung", that repeated what could be done at a paragraph. They were unclear about what they acted on. Now there is one mechanism: every segment has one anchor at its margin, and a click on it *selects* the segment. With the toggle "Mehrfach" on, further clicks add or remove segments, so that several segments are worked on together. What is selected is tinted in the text, and a panel stands at the foot of the notebook for as long as something is selected (Figure 2). The two buttons above the text are gone.

The panel says how many segments are selected and offers the operations on them as a row of tiles; one tile at a time opens its work area. It does not name kinds of segments or lines and does not explain itself; what a tile does stands in its title.

- **Überarbeiten** (rework): an instruction in words, carried out by Famulus (Section 4), or a **preset**, an instruction that is already worded. Several segments of text form one session;
- **Anmerken** (annotate): a note on the selection, kept in a database of the notebook (Section 3.1);
- **Erläutern** (explain): an explanation of a chosen kind, each kind a Partitur that names its context (Section 3.2);
- **Prüfen** (examine): the Sachprüfung, which checks every sentence of the selection against the sources and reports it as *sachlich validiert* or *nicht validiert* ("nicht validiert" says that the evidence does not carry the sentence, not that it is false);
- **Mehr** (more): the source of one segment edited **by hand**; its **history** (Section 5); the **imports** of the project (Section 6); and, on a paragraph of a project that works with the package astra, an **astra object** beside it.

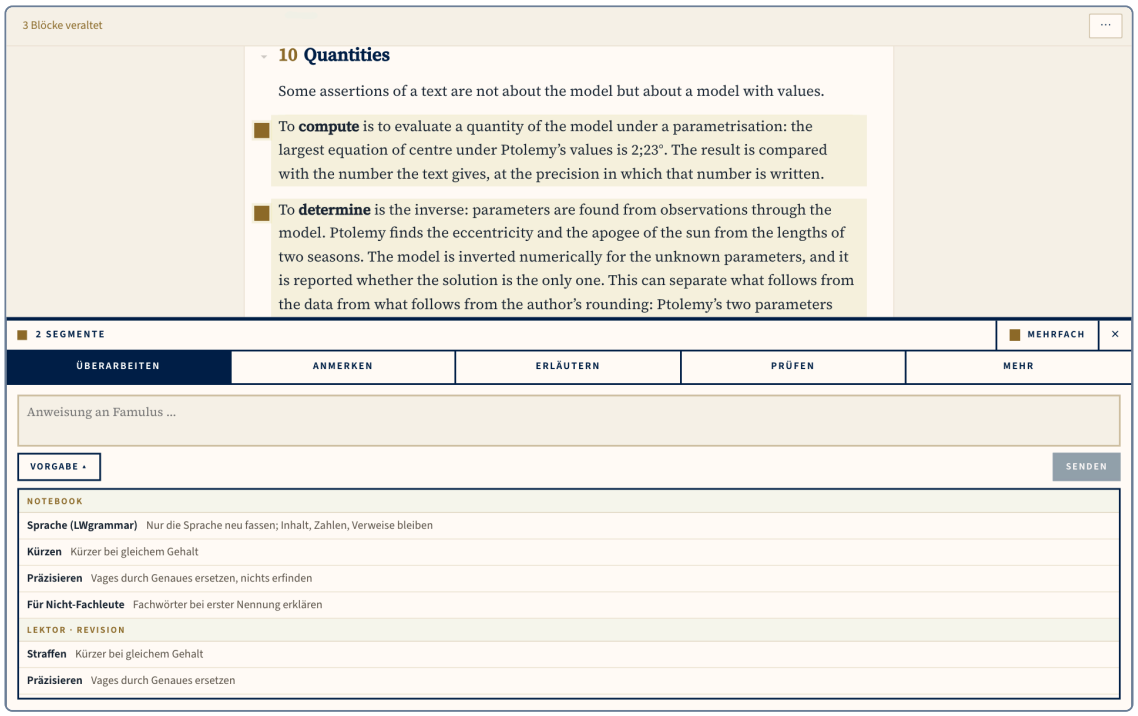


Figure 2 Two paragraphs selected with "Mehrfach" on, and the panel at the foot of the notebook: the count, the toggle, the five operations, and under "Überarbeiten" the field for an instruction with the presets opened (the notebook's own, then the Lektor's by group).

A preset is an instruction that somebody has worded with care once: to shorten without losing a statement, to make vague wording precise without inventing anything, to explain terms for readers outside the field, to check terminology, citations, or the scope of a claim. There are 27 on the installation described here (read from the service on 11 November 2026): 5 of the notebook and 22 of the Famulus Lektor in six groups, two of them in a nested group that the earlier viewer never read. A preset says what kind of segment it is made for; 26 apply to a paragraph, one to a cell's question. Two of them only examine: they change nothing and report a finding. The wording of a preset is taken from the service, never from the page, and what a preset promises to keep (references, citations, numbers, formulas, the end of the sentence) becomes a check of the turn that uses it.

The Sachprüfung was run on another paragraph of the working notebook, the one that introduces Kepler's argument against the equant (11 November 2026). It found seven sentences and reported all seven as *nicht validiert*: they are the author's framing sentences, and the holdings do not carry them.

3.1 Annotations

An annotation is a remark of a reader on one or several segments. It changes no text. The annotations of a notebook lie in a database in the notebook's own folder (`anmerkungen/anmerkungen.duckdb`), so that they travel with the project when it is copied or mirrored (Figure 3). An annotation holds on to its segments

by their wording, not by line numbers. When lines move, it is found again where its text stands; when the text was rewritten, at the block of the notebook most like it, and it then keeps that wording; when nothing like it is left, it is listed “without anchor” and is not dropped. Nothing is deleted: an annotation is active, revised (a later one names it) or revoked. The anchor of an annotated segment is drawn square; the list of all annotations is reached from the menu of the notebook.

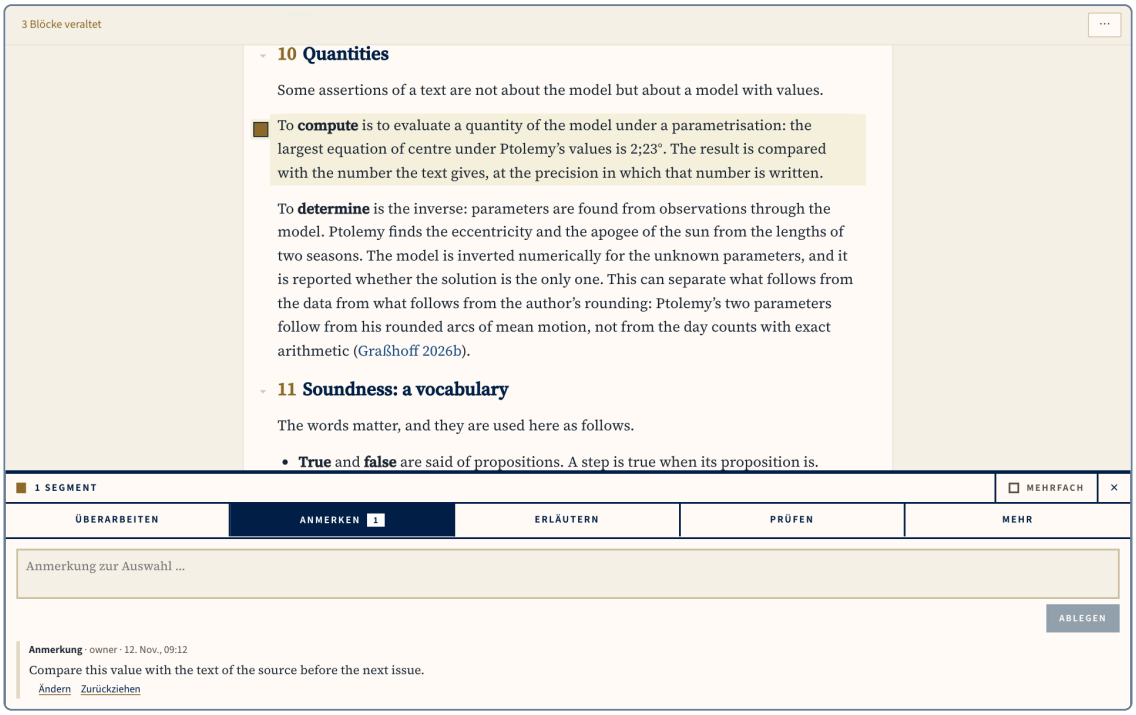


Figure 3 An annotation on one paragraph: the field, and below it the annotation with its author and date; it can be reworded or withdrawn. The annotation of this screenshot was a probe and was removed again.

3.2 Explanations by Partitur

The earlier Erläuterung did one thing: it put the selected text as a question to the evidence space of a holding. A notebook that names no holding with a prepared evidence space, such as the example on triangles that the author opened first, received only a gap. An explanation is now of a *kind*, and each kind is a Partitur: a YAML file that says from which context the explanation is given and what it is to do. The built-in ones are *Zusammenhang* (what the selection says and how it follows from the text before), *Argument*, *Begriff* (the concepts of the selection with their definitions in LWgrammar), *Person*, *Fachlich* (the specialists of the notebook) and *Beleg* (the former Erläuterung). A project keeps its own in its folder, `partituren/erlaeuterung/<kind>.yaml`: a file with the name of a built-in kind takes its place, any other name is a new kind. The panel opens the Partitur of a kind for editing and creates new ones; only a Partitur that can run is written. A Partitur names no model.

The stages of a Partitur name their sources: the selection; the text before it (a number of paragraphs, the section, or the notebook up to the selection); the definitions in LWgrammar of the concepts and persons a first stage finds in the selection; the Famulus Reference Register; the specialists; the texts of imported packages. What “the text before” and “the specialists” mean for a notebook stands in its preamble, under `context:`. A specialist is a package the project imports, or an expert of the platform’s register with its manifest.

The answer is checked against the context it was given. Every statement names the context items it rests on; quotations are compared letter for letter and dropped when they differ; a statement that rests on nothing is marked *ungestützt*. What a source could not give is a *gap* with its reason, shown beside the explanation, and a statement about what is missing rests on that gap (Figure 4).

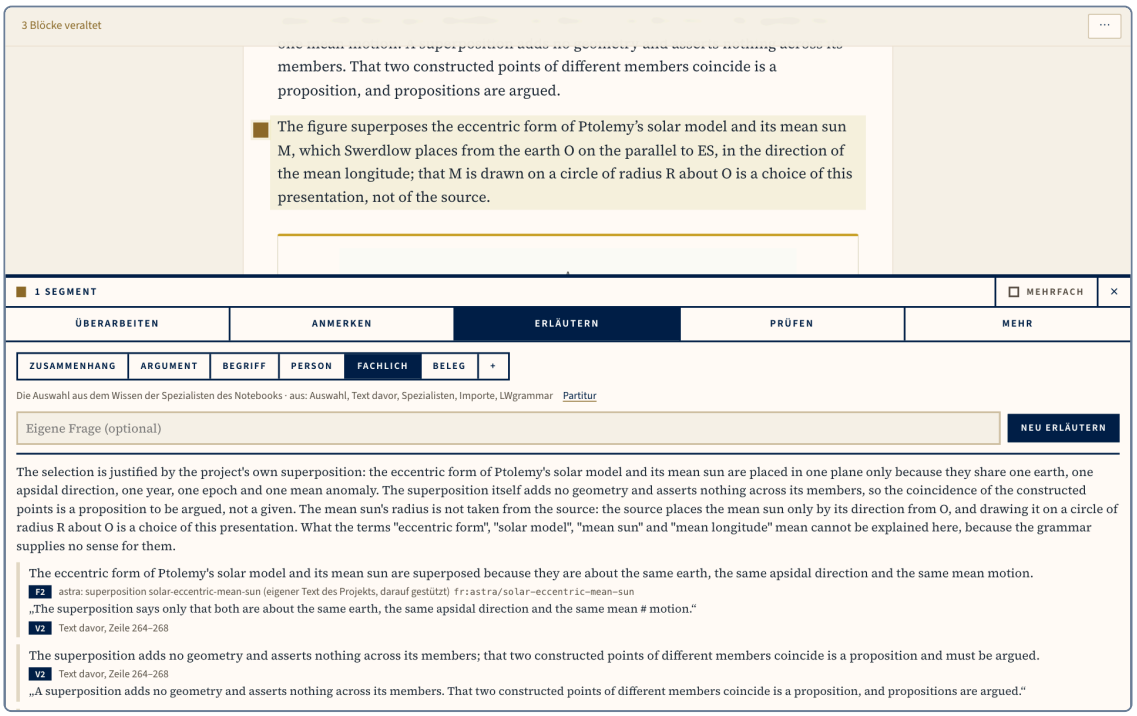


Figure 4 An explanation of the kind *Fachlich* of the paragraph about the mean sun: the kinds as tiles, the answer, and the context that was read — the selection, the section before it, three of the project's own astra texts and the manifest of the astronomy expert.

On 12 November 2026 every kind was run on the working notebook and on the example on triangles, each on DeepSeek in 1 to 5 s (read from the service's answers). *Fachlich* on the paragraph about the mean sun read the figure shown after the paragraph, the superposition and the parametrisation it rests on, all three texts of the project, and the manifest of the astronomy expert. *Person* on the same paragraph found Ptolemy and Swerdlow; the reference register lists several persons under each name, and the explanation says that it cannot tell which is meant. LWgrammar holds no sense for "mean sun", "eccentric form" or "Kosinussatz": each is named as a gap. On the example on triangles, *Beleg* ends with the gap that no holding with an evidence space is named (on the working notebook the same kind, asked of the holding `kepler-prim` on 11 November 2026, had answered with seven statements resting on eight grounds), and *Fachlich* with the gap that there is no specialist for geometry.

3.3 The argument of a selection

The kind *Argument* reconstructs the argument that the selection makes, as it has developed in the notebook up to it: the Partitur reads the whole notebook before the selection. The engine returns statements with a role — premises, steps, one conclusion — and says for each step and for the conclusion from which statements it follows. A premise the text states carries the place that states it and a verbatim quotation; a premise the argument needs but the text does not state is marked as unstated.

No machinery for arguments was built for this. The reconstruction is handed to the prover that the Sachprüfung and the justified answers of the notebook already use: every premise is a unit clause, every step a rule instance, and the prover says whether the conclusion is reached from the premises through the steps named; the proof is then reduced to the clauses that carry it, so that a premise that does no work is shown as idle. The tree is drawn by the notebook's view for proof structures (Figure 5). The check concerns the structure of the reconstruction: that nothing is missing between premises and conclusion and that nothing listed is idle. Whether a step is compelling in content, and whether a premise is true, the prover does not judge; the panel says so.

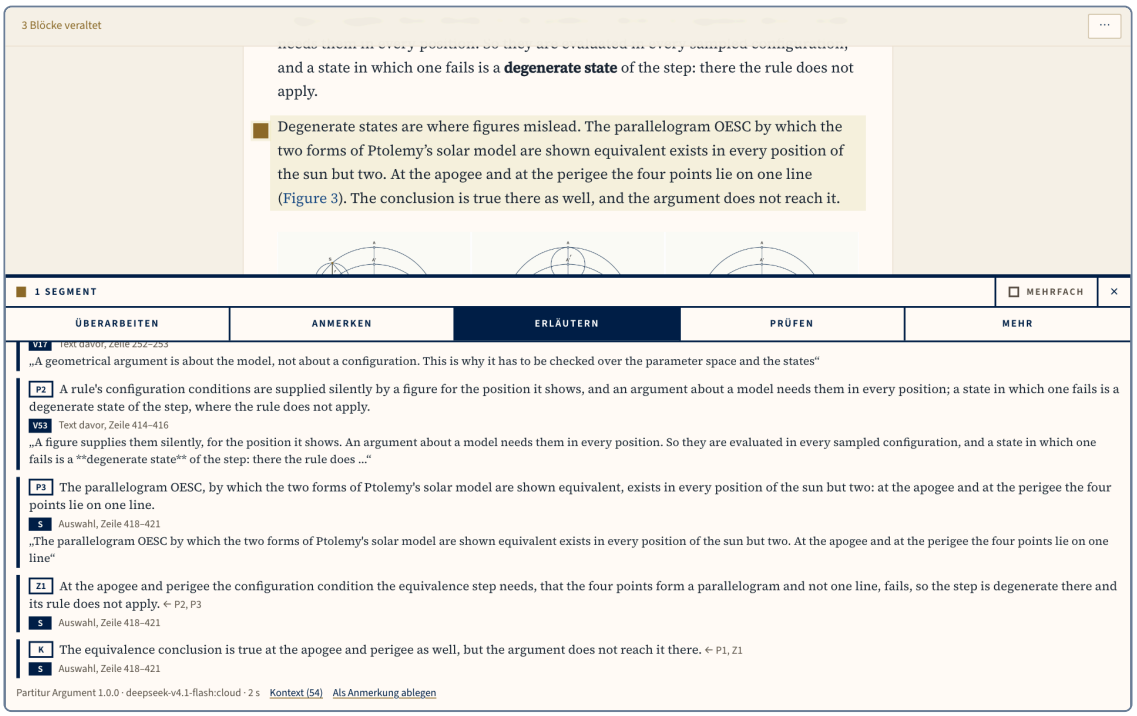


Figure 5 The argument of the paragraph on degenerate states, reconstructed on 12 November 2026: three premises, two of them from earlier sections of the notebook (lines 252–253 and 414–416) with their wording, one step, and the conclusion; the prover reports that the conclusion is reached through the steps named.

4 The instruction session

4.1 The loop

An *instruction session* is opened on one segment. The author gives an instruction; Famulus carries it out; the result is checked and, if it passes, built into the notebook at once, where the author sees it in its place. The session stays open for the next instruction. It ends in one of two ways: **accept**, and the state stays, with the instructions named in the history; or **abort**, and the state before the session stands again, byte for byte, including the removal of files the session created. The last built-in turn can be taken out again on its own. There is one session per notebook at a time; its state is a file, so an open session is still there after the page was reloaded or the service restarted.

A turn does not hand the engine a file to edit. The segment is turned into an *operand*: its **units**, the texts that may be rewritten (the segment's lines; for an astra segment also the project's own astra texts it rests on), its **context**, texts that may be read but not changed, and notes. The engine answers with the complete new text of the units it changes and one or two sentences to the author. Three *handlers* build operands and check proposals, one per kind of segment (Table 2).

Table 2 The three handlers of an instruction session.

handler	segments	units	checks before anything is built in
prose	paragraph, list, quotation, heading, table, figure or table with caption	the segment	the notebook still parses; labels that cross-references use are kept; the rules of usage of LWgrammar; what a preset promises to keep
cell	a cell with its preamble and question	the cell	the fence and a valid preamble; the notebook still parses; a note that the cell must be computed again
astra	a segment that shows an astra object, or a paragraph beside which one is to arise	the segment and the project's own astra texts; new texts may be created	every changed text, and every own text resting on one, passes `famulus-astra check` on all levels of its kind; every object the segment shows is built; own texts name only imported or own texts

4.2 The engine comes from a catalogue

Which engine carries out an instruction is not fixed per feature. The service chooses it from a catalogue by the kind of segment (Table 3). A long source moves the choice up one tier. The reason is plain: tightening a paragraph is routine language work; rewriting a cell needs care with a small formal structure; writing a text in astra/1 that passes six levels of checks is the hardest of the three.

Table 3 The engine catalogue of the instruction session (`standards/engines.yaml`, 11 November 2026).

tier	engine	reached through	starts for
1	DeepSeek V4.1 Flash	the local Ollama	prose, floats
2	Claude Sonnet 5.5	the owner's subscription seat	cells
3	Claude Opus 5.5	the owner's subscription seat	astra segments

A proposal that does not pass its checks is not shown. It goes back to the same engine once, with the wording of the findings; if the mended proposal fails as well, the next stronger engine of the catalogue gets one attempt. If that fails too, nothing is built in and the turn says why. An engine that lacks something the instruction needs is told to change nothing and to say what is missing; two of the runs below show this.

4.3 The runs of 11 November 2026

Table 4 gathers the turns driven on the working notebook during the validation (each turn logged with engine, attempts, time and outcome; the log lies beside the notebook).

Table 4 Turns of instruction sessions on the working notebook, 11 November 2026.

segment	engine	turns	built in	time per turn
astra segment	Claude Opus 5.5	8	6	10.6–64.6 s
cell	Claude Sonnet 5.5	1	1	4.8 s
paragraph, eight presets	DeepSeek V4.1 Flash	8	1	0.6–7.1 s

The figure. The instruction quoted in Section 1 was carried out in one attempt in 28.0 s. The engine wrote a text of the project, `solar-fig-equal-radii-steps-in-aequivalenz`, that draws the same derivation with a line of sight from O to S and five stages, and changed the segment to show it (Figure 6). A second instruction added a stage at the apogee (24.9 s). A third, "let the revolution in the last stage run twice as fast", changed nothing: the engine answered that the figure states no duration, so that it could not halve one, and asked for

the duration in seconds (10.6 s). Given “12 seconds”, it wrote `speed 12 s` (15.8 s). That turn was then taken back, and the session accepted.

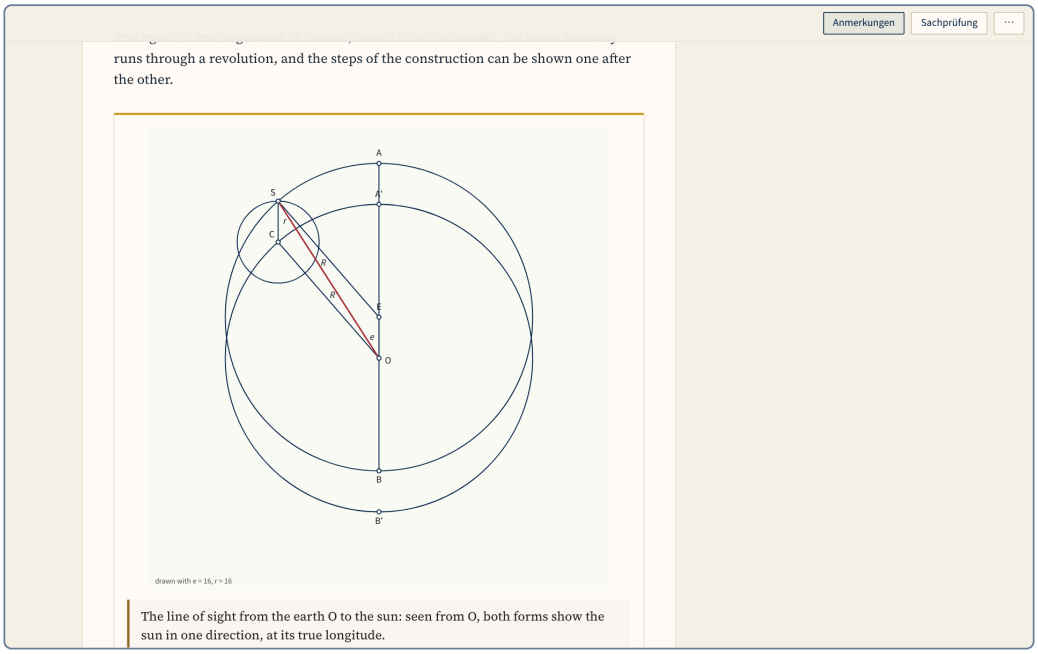


Figure 6 The figure of the equivalence proof after the session, at its fourth stage. The red line is the line of sight from the earth O to the sun; the sentence under the figure belongs to the stage. The figure is drawn from a text of the notebook’s project, not from the shared one.

The paragraph. Eight presets, at least one of every group, were applied to one paragraph in one session. One was built in. In three, the first proposal was refused by the rules of usage (two conflicts with a rule, one lost statement), and the mended proposal returned to the original wording. Four found nothing to change, among them a preset that only examines. The session was then aborted.

Abort. After every aborted session the files of the project were compared with their state before it. In each case all 13 files (the notebook, five astra texts, the project file, the lock, the bibliography, four figures) had the same SHA-256 as before. This was checked for sessions opened by the driver script, through the panel (Figure 7), by commands in the harness, and by the agent in a conversation. A session on several segments writes the changed segments from the lowest upwards and moves every span to where it then stands; on 12 November 2026 a session on two paragraphs was aborted twice through the panel, and the same files had the same SHA-256 again.

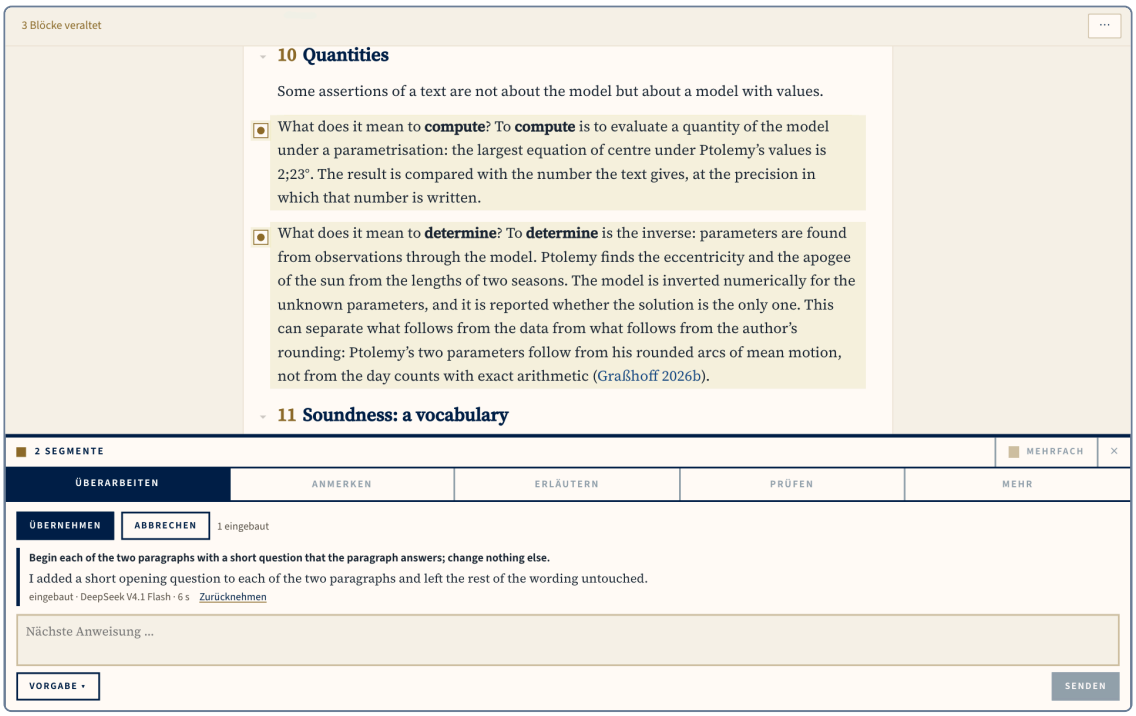


Figure 7 An instruction session on two selected paragraphs (12 November 2026): one instruction, built into both after 6 s, with what the engine said. "Übernehmen" keeps the state, "Abbrechen" restores the state before the session; this session was aborted after the screenshot.

5 Versions of all writers

A notebook has several writers: the reader who edits by hand, an agent's tool, an instruction session, the Lektor when a proposed wording is accepted, an editor outside the service. A change can be taken back only if the state before it was kept. The service therefore keeps a *journal* of every state of a notebook file it gets to see, stored once by content, with the origin of each state: by hand, by Famulus, in a session, a wording accepted, taken back, written outside. A write through the service records the state before and after; reading the history first records the state on disk, so a change made outside appears as such.

The history of one segment is read out of these whole-file states. The segment's lines are followed back from state to state through the alignment of the two texts, and every state in which the segment's text differs is a version of it (Figure 8). Restoring a version writes its text over the segment as a new state, so that a restoration can itself be taken back. The journal lies outside the project folders, which are mirrored between machines; it is the record of one machine.

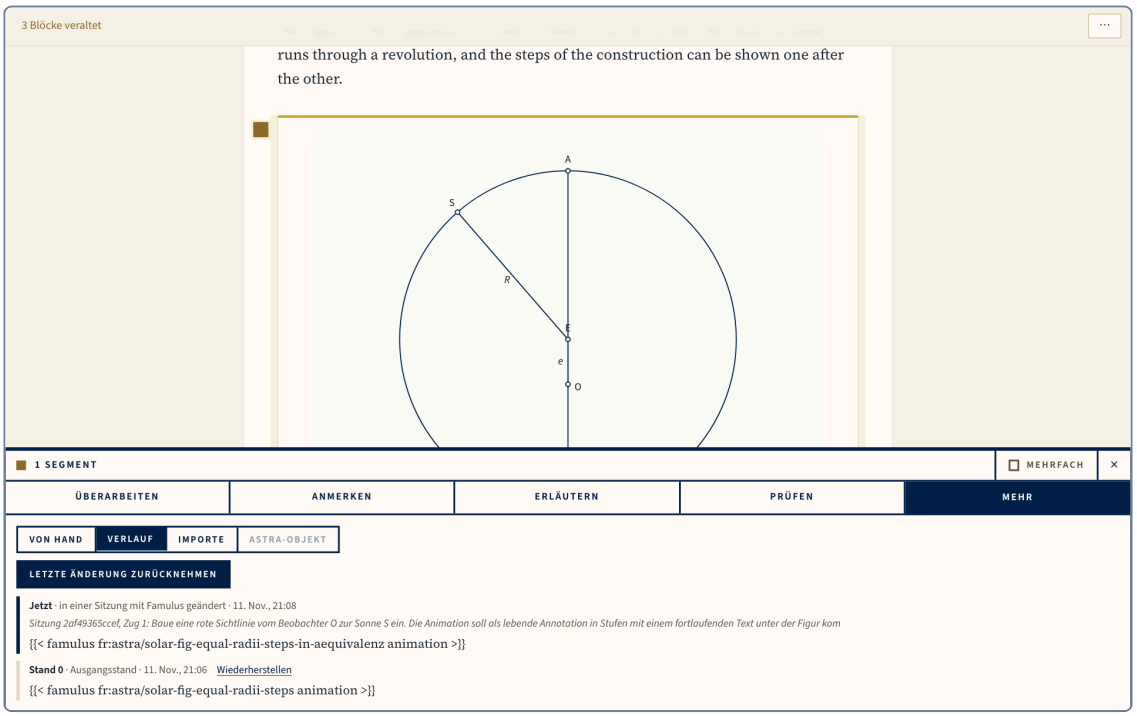


Figure 8 The history of the animation segment under “Mehr”: the state now, written in a session and carrying its instruction, and the ground state. Each earlier state can be restored.

6 Packages, imports, and a project’s own models

6.1 What a project declares

Until now a notebook showed an astra object by naming it, and the whole collection of astra texts was silently available. That is convenient and wrong in the way global names are wrong: nothing says what a notebook depends on, and nothing notices when what it depends on changes. A notebook project now *declares what it imports*, in its project file:

```
imports:
  kepler: fam:kepler-lex-naturae/lex-naturae   # a notebook
  solar:  fr:astra/solar                       # a package, a group, or one text
  heft:   fr:expertise/astra/1                 # an issue of the series
```

The kind of an import follows from its address. A notebook import gives a short name for references to that notebook’s cells and artefacts. An issue import does the same for an imported issue of the publication series. A *package* import is new. A package is a collection of formal texts that a Famulus expert offers with a manifest: for every text its kind, its group, the SHA-256 of its source, and the texts it names. The package `astra` of famulus-astronomica lists 27 texts and one exposition in three groups (11 November 2026: 4 models, 4 parametrisations, 1 superposition, 11 figures, 7 derivations). A project imports the whole package, a group, or one text, and a text always comes with the texts it rests on: importing a figure brings the derivation it draws, the superposition that derivation works in, the models and their values.

Exactly the imported texts and the project’s own texts are available to the project. A segment that shows a text outside the imports is an open reference that names the line which would import it. The project’s lock records the hash of every package text in use; when a text of the package changes, the segments resting on it are *stale* until the author has looked and locked again. Every gap has a name: an address that cannot be imported, a package that is not installed, a group or text the package does not have, a pinned version that is not the installed one, an issue that was not imported, a manifest older than its texts. A project that shows

package texts without declaring the package still reads, because issues published before the declaration existed must keep reading; it is told which line is missing.

The working notebook imports two groups (Figure 9). The package manifest, the declaration, its gaps and the lock are covered by tests in both projects.

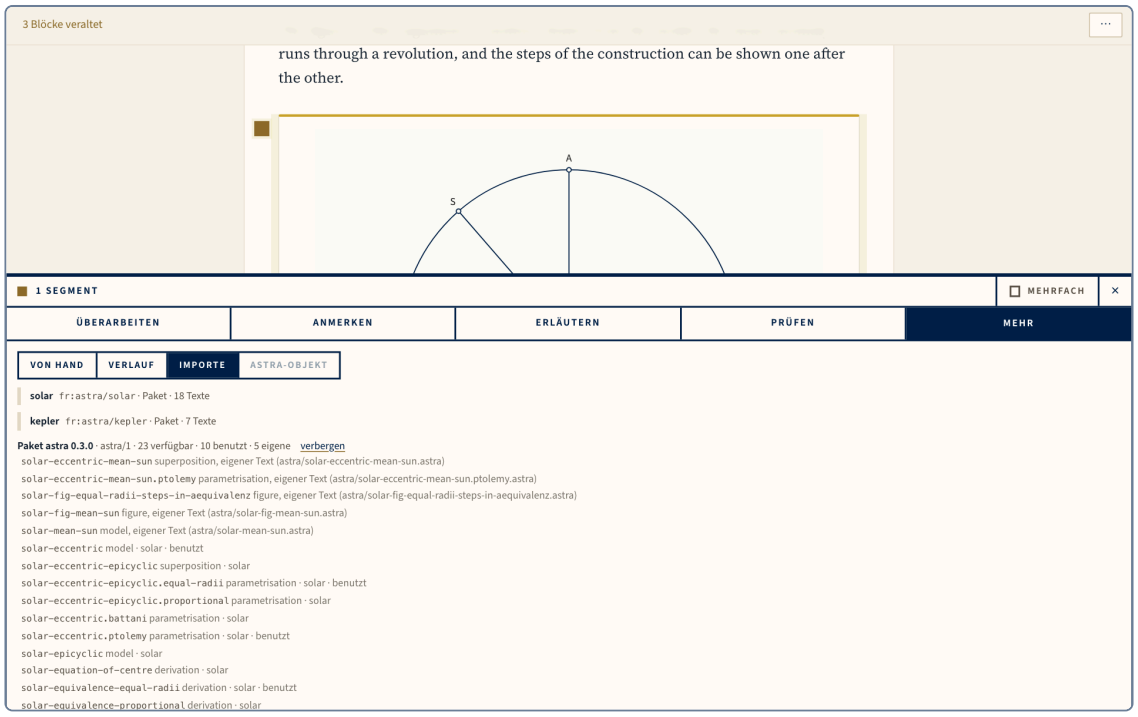


Figure 9 The imports of the working notebook as the panel shows them: two declarations, the package with the texts available, the texts in use and the project's own.

6.2 A new model

A project keeps texts of its own in a folder beside its notebooks. They can be of every kind of the language, and they name imported texts by their identifiers. The figure of Figure 6 is such a text. The harder case is a new *model*. In astra/1 a model is pure geometry and holds no values; values are a parametrisation, each with its source or a stated gap; two models are shown together in a superposition, which identifies their inputs and asserts nothing across them; and a model must be documented: whose model it is, the passage whose use of it is analysed (its *witness*), who wrote the description [3].

On a paragraph about superpositions, the session was opened for an astra object and given this instruction: add a model of the mean sun, moving uniformly on a circle about the observer with the mean anomaly of the eccentric model; superpose it with the eccentric model; show both in an animated figure with the lines of sight to the sun and to the mean sun. The engine wrote nothing. It answered that none of the passages cited by the texts in its context speaks of a mean sun, named the passages it had, and asked for the one that carries the model (22.5 s). That is the intended behaviour: an engine does not compose a source.

The passage exists. Swerdlow writes that "the mean sun, lies on a line parallel to *ES*" in the direction of the mean longitude from *O* [2, 22] (quotation checked against the transcription of scan page 52, paragraph 4, of the edition's copy). It was named in the next instruction, with the remark that the source gives the mean sun only by its direction, so that a circle of radius *R* is a choice of the presentation. In one attempt (64.6 s) the engine then wrote four texts: the model, with that choice stated in its description; the superposition with the eccentric model, identifying earth, apsidal direction, year, epoch and mean anomaly; a parametrisation that takes Ptolemy's values over from the package and marks the radius of the mean sun's circle as a gap, "the source places the mean sun only by its direction"; and the figure. All four passed every level of their kind, and the figure was built and shown under the paragraph (Figure 10). The engine added that it had not seen

the wording of the passage named in the instruction. A following instruction turned the line to the mean sun into a second line of sight (11.7 s).

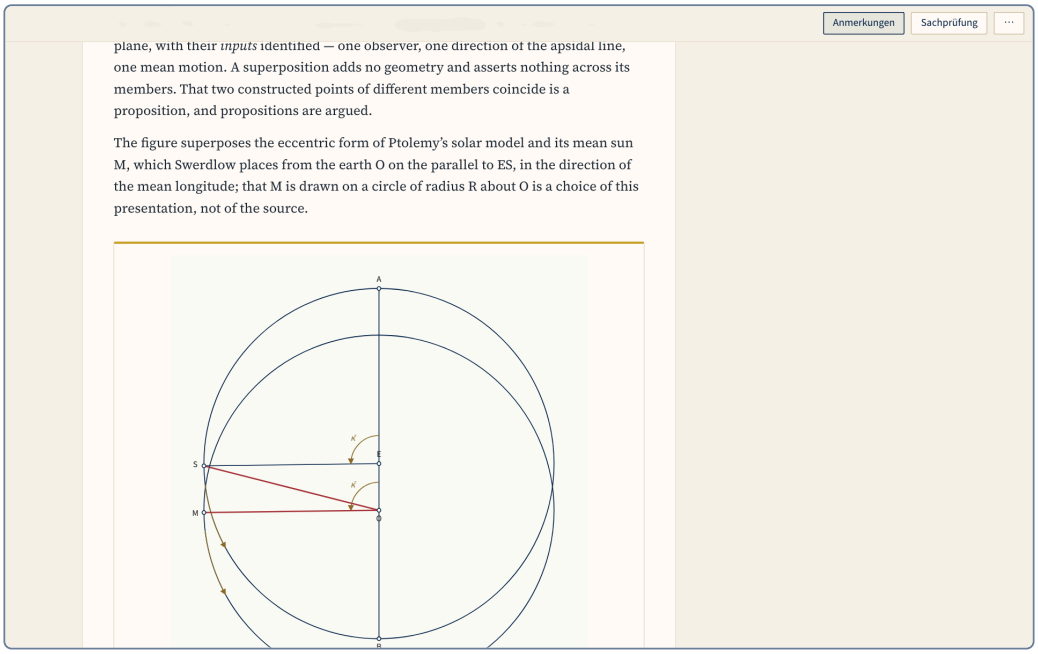


Figure 10 The new model in the notebook: the eccentric form with the mean sun M, drawn from four texts of the project that were written in one turn. The sentence above the figure was written with it and states the choice of the presentation.

Whether the model is a good one is for the author to judge. The notebook says so itself: a section at its end lists what was made in the harness during the validation and has not yet been read by the author.

7 Cells that compute with the package

A cell of a notebook asks a question in words and states in a preamble how it is answered; its result is kept in versions with its artefacts [5]. The issue this notebook was copied from had no cell. Three kinds of step now let a cell ask famulus-astronomica; none of them calls a language model:

- `astra:derive` returns the steps of a derivation as a table, each with its premises, its rule and its status;
- `astra:check` returns the level each of several texts reaches;
- `astra:positions` returns where named points stand at equally spaced states of one revolution, and for two points their distance.

Each returns counted values that an *oracle*, the expectation a cell states about its own result, can be set against. The working notebook has one cell of each kind (Figure 11). The first expects a complete derivation with two givens and three cases; the second that all six texts the argument rests on reach every level of their kind; the third that the sun of the eccentric form and the sun of the epicyclic form are at most 10^{-9} apart over a revolution with the radii of the hypothesis. All three were computed in the harness and passed (the largest distance was 0.0). The table states what was computed with which values. That the two suns coincide for every admissible value is what the derivation shows, not the table.

The third cell was then changed by an instruction, “compute with 36 instead of 24 states and adapt the oracle” (4.8 s). The cell became stale; refreshing the project computed exactly this cell; its artefact got a second version with 36 rows, and the first, with 24, is kept. The texts a step reads count among the inputs of its cell: when one of them changes, the cell is stale.

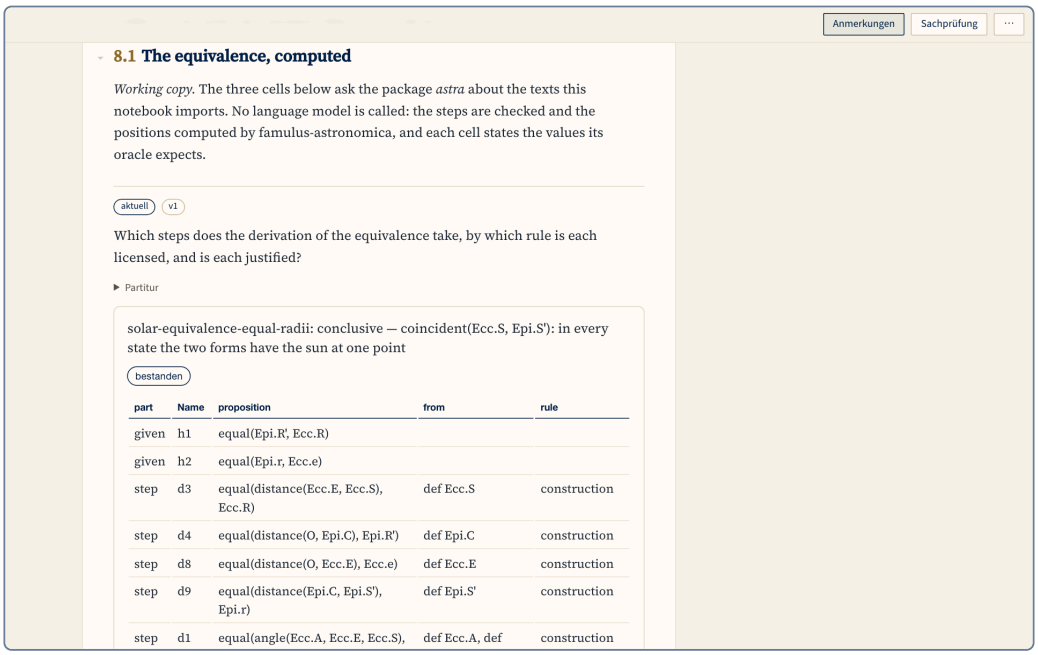


Figure 11 The first of the three cells: the question, the verdict of its oracle, and the table of the derivation's steps with rule and status.

8 Addresses, issues, and an outlook

Every notebook, cell and artefact has an address in the notation famulus-ref/2: `fr:notebook/aequivalenz/geometrical-reasoning-fundamentals` is the working notebook, and `…/sun-distance/places` the table of the third cell. On 11 November 2026 the reference service resolved both, the notebook with its three cells as children. The namespace was entered into the signed register of the node on the same day (snapshot 6, 35 claims, the check without findings).

A published issue of the series Famulus Expertise can be imported into the library as a notebook by its address; the copy keeps the address, version and licence of the original. For this an issue must lay its source open. The exporter of the series now writes, beside the source, the files it names (images, the bibliography, the issue's own astra texts) and a list of them with size and hash, and the import fetches exactly that list. This is built and tested; no published issue has been rebuilt with it yet.

Outlook. A notebook that can be addressed, imported and built upon is a unit of a larger structure: notebooks that cite each other's results, import each other's models, and state by their checks what they have justified. The Famulus documentation calls the perspective of such a network WWK, world-wide knowledge. It is a perspective. What exists is described in this paper: addresses that resolve on one node, imports between projects on one machine, and issues that can be imported from one site.

9 The agent: tools, commands, approvals

9.1 Three ways into the same session

The panel of a selection is one way to give an instruction. In the harness there are two more.

A **command** is typed by the user into the conversation and carried out as typed: `/anweisung` with a place and an instruction, `/vorgabe` to list the presets that apply or to apply one, `/zurueck`, `/uebernehmen`, `/abbrechen`, `/importe`. No model stands between the user and the session, so nothing has to be approved. The answer appears in the notebook pane (Figure 1).

A **tool** is chosen by the agent. `famulus_presets` and `famulus_imports` read; `famulus_instruct` opens or continues a session, with an instruction, a preset, or both, and can open a paragraph for an astra object;

`famulus_session` accepts, aborts or takes back a turn. The tools that write ask the user first. The harness shows a card that says in one sentence what will happen, and the user allows it once or refuses.

9.2 A conversation

The whole path was run on 11 November 2026 in a second instance of the harness with a state folder of its own, on a throw-away copy of the working notebook. The task was given in words: apply the Lektor's preset "Straffen" to the paragraph that begins "Degenerate states are where figures mislead", report what was built in, and ask whether to accept or abort. The agent loaded its skill, read the notebook's outline, found the paragraph, asked for the presets that apply, and called `famulus_instruct`. The card read: Famulus carries out, in this notebook, the preset `lektor:straffen` and builds the result in; a session; aborting restores the state before. It was allowed once. The service built the preset in after 3.6 s. The agent reported three small changes, what had been kept, and a note of the language check, and asked: accept or abort? The answer was "Abbrechen." The agent called `famulus_session` with `abort`; the card said that the state before the session would stand again (Figure 12); it was allowed once; the session was aborted, and the notebook file had the hash it had before the conversation.

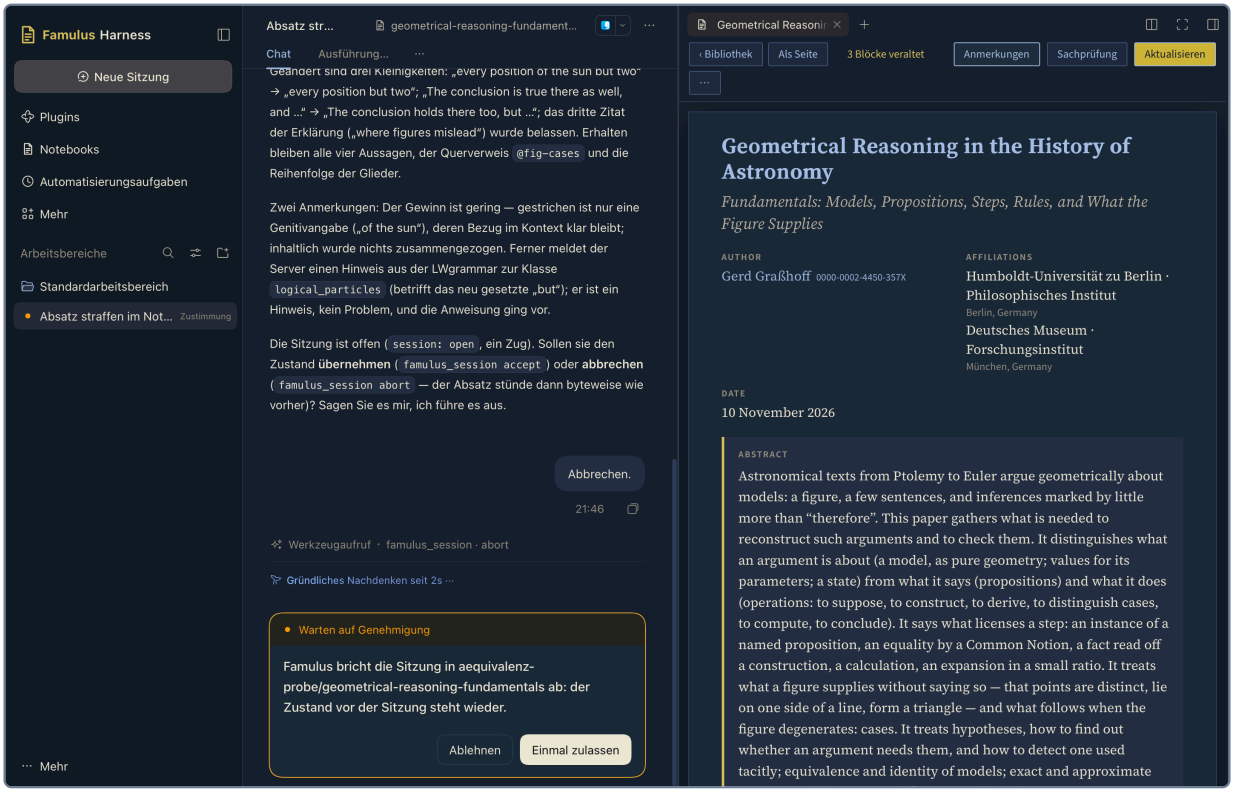


Figure 12 The conversation in the second instance: the end of the agent's report, the user's answer, the tool call, and the card that asks before the session is aborted. The notebook pane on the right shows the copy the conversation worked on.

The approval is per call. That is deliberate: a card names one action on one notebook, and the user who allows it knows what was allowed.

10 Two modes

The harness and the notebook share one palette in two modes: a light mode on a sand ground and a dark mode on a slate blue, not black. Until this validation the astra objects were the exception: in the dark mode they stayed light plates, because their stylesheets set a dark blue as the ground of a band, which the dark mode turns into a text colour. They now take their colours from roles that have a value in each mode, and a figure sits on the dark sheet with its lines, labels and its line of sight readable (Figure 13). In the light mode

the objects are unchanged: screenshots of two arguments and one animation before and after were identical pixel by pixel (11 November 2026). Published issues do not name a mode and stay light.

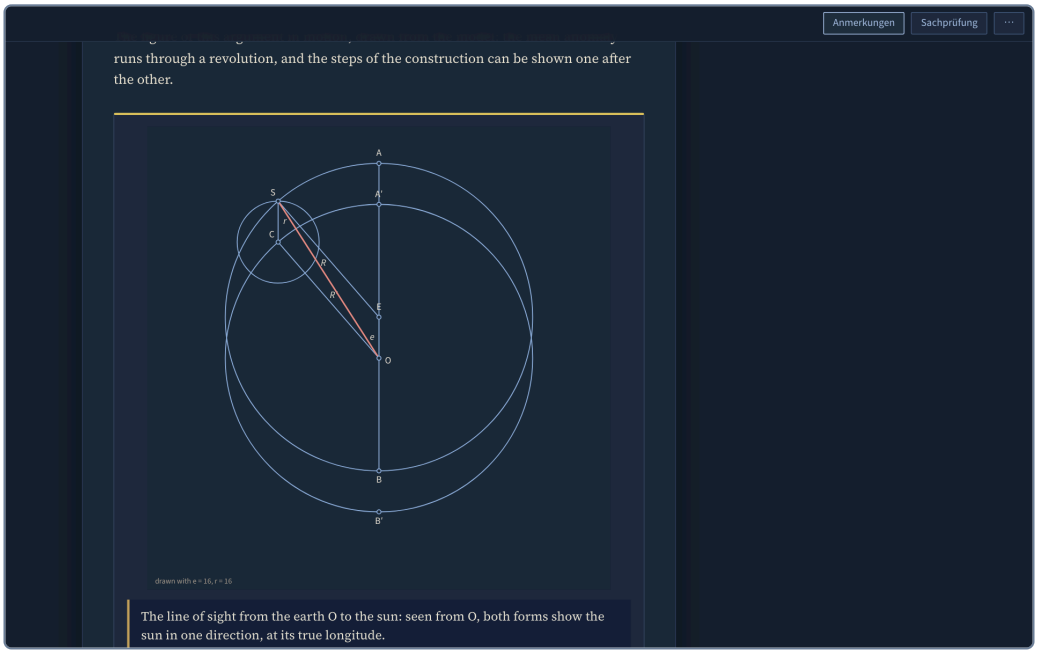


Figure 13 The narrated figure of Figure 6 in the dark mode.

11 What is not built, and the defects we met

Only what is described above runs. The following is not built, or was found defective during the validation of 11 November 2026.

Not built.

- *Published issues.* No published issue has been rebuilt with the list of its open files, and none carries a narrated figure. Release and seal of an issue are steps of its author.
- *One package.* `astra` is the only importable package. Other Famulus experts offer services, not packages of texts.
- *Own astra texts at the reference service.* The service resolves the texts of the package. A project's own astra text is decided by the notebook service; its address does not resolve elsewhere.
- *Versions as addresses.* The version journal is a record of one machine. Its states are not `fr:` versions and are not shared between machines.
- *A specialist for geometry.* The notebook on triangles names one; none exists. Experts of the register give their manifest (subject, grounds of their gates, what they do not cover); an explanation does not yet ask an expert a question of its own.
- *Definitions.* LWgrammar holds few senses for the terms of the two notebooks; most lookups end as gaps.
- *The content of a step.* The prover checks the structure of a reconstructed argument, not whether a step is compelling. The reconstruction itself is the engine's reading and has not been compared with a reading by the author.
- *Several segments.* A session on several segments takes text only; a cell or an astra segment is worked on alone.
- *One session, one author.* A notebook has one open session; two authors cannot hold sessions on one notebook.
- *Presets for figures and cells.* Of the 27 presets, 26 are made for prose. An astra segment is offered one, a cell two.
- *Escalation observed.* In the runs of Table 4 a mended proposal always sufficed or returned to the original. The step to the stronger engine is covered by a test and was not observed on the working notebook.

- *Editing by hand* was not driven separately on the working notebook; the path is covered by tests.
- *Approval for a whole session.* Every writing call of the agent asks. A session of ten turns led by the agent means ten cards.

Defects found during the validation.

1. *A command's answer was invisible.* On the start screen of a new conversation the harness page showed no answer to a command; the command ran, the list of presets it should have shown did not appear. The answer is now sent to the notebook pane (Figure 1). Fixed.
2. *A read asked for approval.* Reading the status of a session went through the gate of the writing tool it belongs to and produced a card. Only accept, abort and take-back ask now. Fixed.
3. *Quiet buttons drawn as boxes.* In the window of a segment the entries of the list and the links were drawn with the border of ordinary buttons. Fixed.
4. *The language of an answer.* One answer to a German instruction came in English, the language of the notebook. A line of the prompt now asks for the language of the instruction; this was not measured again.
5. *A figure lags behind its turn.* After a turn on an astra segment the window says "built in" a few seconds before the rebuilt figure appears, with nothing to say that it is being rebuilt. Not fixed.
6. *A broken build.* While the dark mode was built, the viewer served for about five minutes a build in which pages with a network artefact stayed empty. Fixed.
7. *A stuck deployment.* The publication of the astra documentation stayed in the state "building" at the hosting service and had to be requested a second time.
8. *A window that explained itself.* As validated, the window of a segment opened with a paragraph of instructions for use, a heading that named the kind of segment and its lines, and a list of operations with a sentence each. The author, on first use, called it a novel. It was reduced to a field and a toolbar on the same evening. Fixed.
9. *The same thing in two places.* The bar above the text still offered "Anmerkungen" and "Sachprüfung" beside the window of a segment, a plain annotation had no entry point at all, and an explanation knew one kind and one source. On 12 November 2026 the buttons were removed and the window became the panel of Figure 2. Fixed.
10. *A theorem read as a person.* Asked for the persons of a paragraph that names none, the engine once listed "Kosinussatz". The instruction of that stage now says what a person is, and a kind that is about persons ends with a gap when the selection names none. Not measured again beyond the runs reported.
11. *The text lags behind a turn.* After a turn or an abort the page reads the notebook again in two to three seconds and shows nothing meanwhile. Not fixed.

The changes made to the working notebook during the validation have not yet been read by its author. They are listed in the notebook, each with the instruction that made it, and each can be taken back.

12 Contribution to the thesis

The lead thesis of the Famulus papers says that Famulus gives historical research a new quality through documented evidence, a justification for every Aussage by the standards of its discipline, integration into a research group, and learning that turns corrections into rules. The harness contributes to the third pillar: the notebook becomes the object a group works on, with an address, declared imports, and a history that names every writer. It shows the second pillar on the scholar's own text. A change proposed by an engine is treated as a claim to be checked, by the notebook's grammar, by the rules of usage, by the checker of the formal language it is written in, before anybody sees it; where the ground is missing, as with the witness of the new model, the engine says what is missing and writes nothing. The harness does not decide whether the new model is right, or whether the tightened paragraph is better. It makes sure that what the author judges has passed its checks, says where it came from, and can be undone.

Bibliography

[1] G. Graßhoff, "Geometrical Reasoning in the History of Astronomy. Fundamentals," 2026.

[2] N. M. Swerdlow, *The Renaissance of Astronomy*. in Sources and Studies in the History of Mathematics and Physical Sciences. Springer Nature Switzerland, 2026. doi: 10.1007/978-3-031-84760-8.

[3] G. Graßhoff, "astra/1. A Language for Historical Astronomical Models and the Geometrical Reasoning about Them," 2026.

[4] G. Graßhoff, "Famulus. Eine verteilte lokale Plattform für professionelle Forschung," 2026.

[5] G. Graßhoff, "Famulus Notebook. A Teaching Book: Writing, Running, Checking, Annotating, Sharing and Publishing Scholarly Notebooks," 2026.