DI∀L∃CTIC: A Practice Environment for Machine-Checked Argument Reconstruction

Stefano M. Nicoletti[1,2⋆] and Edoardo Putti[3]

[1] Department of Pure and Applied Sciences, University of Urbino "Carlo Bo", Urbino, Italy stefano.nicoletti@uniurb.it
[2] Semantics, Cybersecurity & Services, University of Twente, Enschede, the Netherlands
[3] Formal Methods & Tools, University of Twente, Enschede, the Netherlands
e.putti@utwente.nl

Abstract. Philosophical arguments are traditionally exposed to dialogic iteration and improvement cycles, because often the most consequential steps are the one that ordinary prose makes easiest to miss. Scope, underspecified premises or hidden inferential steps can decide whether a conclusion follows, but natural language often hides these away. Proof assistants – on the other hand – offer users the opportunity to mechanize arguments, improving clarity and transparency of inferential steps. However, these systems often present a steep learning curve. To address this gap, we present Dialectic, a Controlled Natural Language (CNL) framework based on the Lean theorem prover, that makes argument reconstruction readable, editable, and machine-checked. Students learning formal argumentation link claims to philosophical sources, state their formal meanings, write visible inference steps, revise assumptions, compare readings, and record objections without interacting with underlying Lean terms or tactics. We ease the learning curve for students by providing users with inference suggestions for their goal, and diagnostics on inferences, all within the Dialectic notebook. Dialectic currently supports constructive, modal, and counterfactual argument forms with explicit verification boundaries. Lean then checks logical entailment under each reconstruction while interpretation remains open to transparent philosophical scrutiny, making the interface between philosophical interpretation and argument correctness explicit and checkable. Dialectic thus offers a pedagogical platform for students to move towards formal representations of their philosophical arguments, without requiring expert knowledge in interactive theorem proving or programming skills.

Keywords: Controlled natural language · argument reconstruction · philosophy education · Lean · proof assistant

1 Introduction

Philosophical arguments improve when their commitments can be exposed, challenged, and revised. Yet ordinary prose often hides key ingredients of an

⋆ The authors contributed equally and are listed in alphabetical order.

argument: the scope of a claim, an unstated premise, or a shift from possibility to necessity. Formalization can aid in uncovering these choices, and in mechanically proving that an argument is correct. Proof assistants – like Lean [18], Isabelle [21], or Rocq [1] – aid users in achieving this goal interactively, but the learning curve of these instruments is often steep. *Controlled Natural Language (CNL)* front-ends for proof assistants offer an appealing alternative with natural-language-like syntax that retains formal semantics [14,13]: e.g., Yalep and Verbose Lean offer domain-specific CNLs to ease new users and students into mathematical proofs [16,25]. What remains missing is a practice environment for philosophy students in which they can learn argument reconstruction with the support of an interactive prover, but without having to be challenged by its steep learning curve.

Our approach. We present Dialectic, a CNL and notebook environment for machine-checked argument reconstruction based on Lean. The notebook interface offers a structured environment in which to write and analyse philosophical arguments. Students explicitly reference the argument in the source text they want to analyze, offer a transparent paraphrasis of key sentences and use Dialectic's CNL to mechanize and check its correctness. These reconstructions can then be challenged with alternatives/oppositions and automatically rechecked for validity. Learning is fostered by Dialectic's interactive nature: e.g., incomplete deductions receive goal-directed suggestions, and reconstructions are supported by targeted diagnostics, like inconsistency or non-entailment checks.

Summary of contributions:

1. A CNL supporting the practice of *philosophical argumentation*, grounded in the *Lean theorem prover*;
2. a notebook environment to support four connected reasoning activities: i.e., sources *interpretation*, *deduction*, *alternative* reconstruction, and argument *diagnosis*; and
3. *interactive support* for incomplete deductions, together with machine-checked outcomes for deduction rejection, inconsistency, and non-entailment.

Structure of the paper. Section 2 presents a list of requirements and discusses related work. Section 3 discusses Dialectic's reasoning workflow, intended audience, and potential coursework applications. Section 4 introduces the notebook interface through a running example, and Section 5 covers interactive deduction, alternatives, and diagnostics. Section 6 describes the supported logical fragments and their boundaries. Section 7 discusses implementation details and Section 8 presents limitations and future work.

2 Requirements and Design Rationale

2.1 Requirements for reconstructing an argument

The first step of formal argumentation is interpretation and reconstruction in a logical framework: this includes deciding what a passage says, which premises it supplies, and which inferences are allowed on these premises. Michels separates the choice of inferential support, formal language, adequacy criteria, and logic in argument formalization [17]. Work on computational hermeneutics likewise treats formalization as a comparison among explicit, theory-relative interpretations rather than a single automatic translation [8]. These observations guide our (non-exhaustive) requirement elicitation:

R1: Readable surface. The interpretation given by Dialectic's users must be readable without prerequisite Lean knowledge.

R2: Source traceability. Each reconstructed claim must identify the delimited source unit that motivates it. Paraphrase and formal meaning must remain separately inspectable.

R3: Controlled meaning. Every phrase used for checking must have one formal interpretation in the selected logic profile.

R4: Visible inference. The user must name the premises and result of each deductive move. Automation may propose a move, but it must not hide an inferential step.

R5: Explicit alternatives. A competing reconstruction must state its changes and recheck the preserved deduction in a fresh environment.

R6: Negative evidence. Rejection of one deduction, witnessed inconsistency, and countermodel-supported non-entailment must be distinct results.

R7: Embedding (VS Code + Lean). Parsing, checking, deduction state, and editor actions must work in an ordinary Lean file with the standard VS Code extension.

GenAI-assisted design and implementation. We use generative AI to develop Dialectic. In our development, we tested the interactions, reviewed generated proposals and assessed their implementation. The entire implementation qualifies as *untrusted*, i.e., it cannot check proof terms: in other words, only the Lean kernel can verify that a given term is a proof for a proposition and our implementation only translates between the surface language of Dialectic and Lean's language.

2.2 Related work: formal methods & LLMs, controlled languages and (mechanised) argumentation

Recent autoformalisation research provides a related use of language models in formal methods. Wu et al. evaluates translating mathematical statements into Isabelle/HOL [28], while Murphy et al. combine language models with symbolic reasoning to formalise geometry in Lean [20]. These contributions explore one

direction at the intersection between formal methods and LLMs, i.e., constraint LLM output via the guarantees offered by symbolic reasoning/interactive theorem proving. Our contribution, instead, addresses the usability problem of these formal reasoning tools, and we leverage AI-assisted coding to derive a human-checked implementation from high-level requirements, via a controlled natural language. Controlled natural languages trade unrestricted expression for predictable interpretation. Kuhn compares such languages by precision, expressiveness, naturalness, and simplicity [14,13]. Previous literature prove that it is possible to provide an effective CNL for the Lean language, but focus on mathematical proofs: e.g., Verbose Lean demonstrates that a proof assistant can support paper-like proof language in mathematics [16]. Yalep, instead, places a small declarative layer above Lean so that high school students can work on mathematical proof without first learning prover syntax [25]. Dialectic adopts their separation between a task-facing language and its checked elaboration, but extends it to philosophical argumentation, reconstructions and disagreements [4]. Argdown offers a complementary precedent for lightweight textual representations of pro and contra reasoning [27]. Dialectic incorporates these reasoning patterns via the explicit encoding of formal interpretations of given sources, reconstruction of deductive steps and alternative/opposing reconstructions of the same argument. More literature substantiates our claim w.r.t. the usefulness of provers in history of philosophy [5], and in understanding/mechanisation of philosophical arguments/computational philosophy [7,23].

3 Dialectic Design and Use Case

Reasoning activities Dialectic is designed so that four connected reasoning activities – encoding/interpretation of the premises, argumentation/deduction, opposition, and diagnosing the argument – reside in the same notebook. We draw on computational hermeneutics, which treats formal reconstruction as an iterative process of interpreting passages, checking their inferential consequences, and assessing their role in an argumentative context [9]. The opposition component also reflects the dialectical perspective on argumentation, in which arguments are considered through challenges and responses [4].

Envisioned workflow Students begin by deciding what their argument is about: they name the domain, its predicates, and the aliases used to read them. To make the reconstruction accountable, students explicitly link natural-language passages in the source manuscript to specific formal claims in Dialectic's CNL (*interpretation/encoding*): a paraphrase records their reading, while its formal meaning in the CNL clarifies the formal commitments that will be translated into a Lean term, which can be mechanically checked. Named deduction steps then become the building blocks of an argument that students can inspect and revise (*deduction*). Changing an assumption lets them ask which steps survive a different reading and respond to objections or competing reconstructions (*opposition/alternative*). Where supported by the selected logic profile, Dialectic checks

contradiction witnesses and countermodels, giving students concrete evidence of inconsistent assumptions or non-entailment (*diagnostics*). These activities need not follow a fixed sequence. A countermodel may prompt students to reconsider a premise, while an objection may reveal a commitment that their first reconstruction left implicit. The notebook keeps these revisions connected to the source and separates the questions they address: whether the reconstruction adequately represents the passage, whether the written steps establish the conclusion, whether the assumptions are jointly inconsistent, and whether they can hold while the conclusion fails. Formal checking addresses the inferential and semantic questions supported by the profile. Students remain responsible for defending their interpretation and assessing the truth or acceptability of its premises.

Intended audience and coursework applications We therefore position Dialectic as a tool for introductory reconstruction exercises in introductory logic, introductory philosophy, and interactive philosophy seminars. This positioning follows established teaching practices found in introductory courses to logic/argument reconstruction. For example, Oxford's *Introduction to Logic* draws and explores the connection between formal inference with arguments expressed in ordinary natural language [26], while students in MIT's *Logic I* translate English sentences into formal notation, construct deductions, and test whether the premises can be true while the conclusion remains false [10]. Stanford's argument-reconstruction materials further aligns with our intent, in exploring readings of Aristotle, Kant, and Perry via explicit reconstructions and objections [3]. For selected exercises in philosophical logic, Dialectic's modal profile also connects with topics taught in courses such as MIT's *Modal Logic*, including necessity, possibility, possible-world semantics, and validity testing [29]. These curricula provide precedents for the intended tasks and thus inform the usage scenarios envisioned for Dialectic.

A practical example. A Dialectic notebook is best understood through an example exercise: *Corroborated Testimony* (see Section 4.1). This simulates a classroom scenario that links independent corroboration, responsible assessment, credibility, and prima facie warrant. Its purpose is to expose a two-step deduction and a different interpretation under a quantifier change. This completes the exercise and showcases all reasoning steps. The notebook (Codes 1.1 to 1.4) showcases our reasoning schema, moving from vocabulary and source-linked meanings (Code 1.1) to named deduction steps (Code 1.2), one explicit revision (Code 1.3), and interpretation of the resulting diagnostics (Code 1.4). The notebook is interactive – just like the Lean theorem prover – and *partial notebooks* are supported. As an example, when opening an empty reasoning *Step*, students are presented with deterministic suggestions on how to pursue the argumentative goal. The same notebook can be used for competing reconstructions and to check the same deduction under both. This allows the user to compare the validity of a deduction when a premise change.

4 The Notebook interface

In the following we introduce the Dialectic notebook interface through a running example. The example shows both the syntax of the tool and how it supports the users in the formalization of philosophical argumentation. The notebook interface fixes a grammar for the formalization structure and the shared input CNL.

4.1 A running example

Codes 1.1 to 1.3 give the complete executable notebook for *Corroborated Testimony*. Together, they follow the four reasoning steps introduced in Section 3. Code 1.4 further shows the diagnostic output of the final recheck.

Encoding and interpreting the premises. First we choose an appropriate logic profile for the argument reconstruction (further details are given in Section 4.2). Then we declare the vocabulary needed for the reconstruction and fix source *sentences* that are relevant. Each *claim* links a natural language sentence to an explicit *paraphrase* and to one *formal meaning*.

Code 1.1. Encoding and interpretation of the premises.

```
import Dialectic.Language

Argument CorroboratedTestimony

Logic profile CoreLogic
End logic

Vocabulary
  Type Report called report
  Predicate independentlyCorroborated describes Report
  Predicate responsiblyAssessed describes Report
  Predicate credible describes Report
  Predicate primaFacieWarranting describes Report
End vocabulary

Source WorkshopScenario
  Citation "Invented classroom scenario for Dialectic"
  Sentence 1 "Every report supported by independent sources is responsibly
     assessed."
  Sentence 2 "Every responsibly assessed report is credible for the purpose of
     this exercise."
  Sentence 3 "Every credible report provides prima facie warrant for belief."
  Sentence 4 "Every independently corroborated report is credible."
  Sentence 5 "Every independently corroborated report provides prima facie
     warrant for belief."
End source

Reconstruction Original
  Claim P1 from WorkshopScenario sentence 1
    Paraphrase "Every independently corroborated report is responsibly assessed.
     "
```

```
    Formal meaning Every independentlyCorroborated report is responsiblyAssessed
  End claim

  Claim P2 from WorkshopScenario sentence 2
    Paraphrase "Every responsibly assessed report is credible within the
     exercise."
    Formal meaning Every responsiblyAssessed report is credible
  End claim

  Claim P3 from WorkshopScenario sentence 3
    Paraphrase "Every credible report provides prima facie warrant for belief."
    Formal meaning Every credible report is primaFacieWarranting
  End claim

  Claim CredibilityBridge from WorkshopScenario sentence 4
    Paraphrase "Every independently corroborated report is credible."
    Formal meaning Every independentlyCorroborated report is credible
  End claim

  Claim Conclusion from WorkshopScenario sentence 5
    Paraphrase "Every independently corroborated report provides prima facie
     warrant for belief."
    Formal meaning Every independentlyCorroborated report is
     primaFacieWarranting
  End claim
```

Argumentation and deduction. Within the *Original* reconstruction of the chosen argument, the student names the claim that they want to argue for – in this case it's *Conclusion* – then they name the intermediate claims and construct the inference steps that lead to the final argumentative objective. Lean checks each step of the deduction for correctness under the specified logic profile.

Code 1.2. Argumentation by two named deduction steps.

```
  Deduction TestimonyWarrant
    Goal Conclusion
    Step CredibilityFollows
      From P1 and P2 conclude CredibilityBridge
    Step ConclusionFollows
      From CredibilityBridge and P3 conclude Conclusion
  End deduction
End reconstruction
```

Opposition through an alternative reconstruction. After the first reconstruction, students can present an *Alternative*. In the example notebook, the objection weakens P1 from a universal to an existential claim, using the *Change* and *Alternative meaning* keywords. The alternative retains the original deduction so that Lean can recheck its steps under the changed interpretation via the *Recheck* command. A sketch area is offered to record a natural language explanation of the objection, under *Objection*.

Code 1.3. An opposing interpretation of P1.

```
Alternative LimitedCorroboration based on Original
  Change P1
    Original meaning Every independentlyCorroborated report is
     responsiblyAssessed
    Alternative meaning Some independentlyCorroborated report is
     responsiblyAssessed
  End change

  Recheck the same deduction TestimonyWarrant

  Objection ScopeOfCorroboration about P1
    Statement "The exercise may support responsible assessment for one assigned
     report without supporting the universal premise used by the original
     deduction."
  End objection
End alternative

End argument
```

Diagnosing the argument. In this example Lean certifies that the original deduction is correct and is thus accepted. The recheck, however, rejects `CredibilityFollows`, because the alternative P1 does not provide the universal premise required.

Code 1.4. Diagnostic output for the alternative reconstruction.

```
Original reconstruction: TestimonyWarrant accepted
Alternative LimitedCorroboration: TestimonyWarrant rejected
  Failing step: CredibilityFollows
  Changed claim: P1
  Reason: the alternative supplies an existential claim,
          but the step requires the original universal claim.
```

4.2 Notebook and profile

The notebook interface provides the structure for the formalization and separates it from its logical interpretation. The structural elements (*claim*, *source*, *paraphrasis*, *deductions*) contain the references to the rest of the notebook elements, e.g. a *claim* contains a reference to premises and conclusion. The profile provides the logical interpretation for each of these structural elements. In Code 1.1 the CoreLogic profile is chosen as the target of the encoding. CoreLogic meanings map to constructive Lean propositions. For example, the sentence `Every credible report is primaFacieWarranting` maps to a universal implication between unary predicates. The deduction steps in Code 1.2 are mapped to the composition of the respective deduction steps in Lean. As of today, mismatches between the syntax allowed by a given profile and what the students write are caught before checking the deductions and diagnosed (see Section 7.2).

Translating the running example For the running example, the checked CoreLogic context contains a type *Report*, four unary predicates, and the three premises

$$P_1 : \forall r,\ corroborated(r) \to assessed(r),$$
$$P_2 : \forall r,\ assessed(r) \to credible(r),$$
$$P_3 : \forall r,\ credible(r) \to warranting(r).$$

5 Using the notebook

5.1 Interactive notebooks

Lean is an interactive theorem prover capable of providing feedback on the proof state as the user types. Dialectic is implemented in Lean itself and builds upon this interactive feature to present the state of a reconstruction in the same controlled language (see Section 7.2). This also extends to the deduction state where premises appear when selected along with the intended conclusion. This feature allows the user to read and write the same language while – unbeknownst to them – the notebook and profile translate the Controlled Natural Language to the Lean language and the Lean system checks the deductions. This extends to the hints and diagnostic that a profile may implement (see Section 5.3). As an example consider the partial deduction in Code 1.5:

Code 1.5. A deduction awaiting its first step.

```
Deduction TestimonyWarrant
  Step
End deduction
```

At `Step`, the Infoview panel (see Section 7.2, not shown here) shows *Choose the next deduction step*. The panel lists the selected profile, declared claims, source references, and moves implemented for the current fragment. A stock *Quick Fix*[4] replaces the exact marker range with an explicit `Goal` and closes the step.

Code 1.6. A checked prefix awaiting the final step.

```
Deduction TestimonyWarrant
  Goal Conclusion
  Step CredibilityFollows
    From P1 and P2 conclude CredibilityBridge
  Continue deduction
End deduction
```

Depending on the logic profile chosen the outcome is different. CoreLogic proposes the step in Code 1.7. ModalK proposes only box modus ponens, and the Counterfactual profile proposes only its shared-selection consequence.

[4] For further info on *Quick Fix* see https://code.visualstudio.com/docs/editing/editingevolved#_code-action and https://lean-lang.org/doc/api/Lean/Server/CodeActions/Basic.html.

Code 1.7. The next CoreLogic step.

```
Step ConclusionFollows
  From CredibilityBridge and P3 conclude Conclusion
```

5.2 Alternative reconstruction

Dialectic is aware of the mode in which it is used. E.g., in an alternative reconstruction the system provides additional diagnostics for the following cases: counterexample and *non-entailment* (see Section 5.3). As an example of an alternative reconstruction we provide a `CounterfactualMatch.lean` notebook, describing a counterfactual scenario on lighting a match. Code 1.8 gives the source and its original reconstruction and then presents an alternative, where we change the first conditional, and recheck the same deduction:

Code 1.8. An alternative reconstruction on a counterfactual scenario.

```
Source LewisMatchScenario
  Sentence 1 "Had the match been struck, it would have lit."
  Sentence 2 "Had it lit, its flame would have been visible."
  Sentence 3 "Had the match been struck, its flame would have been visible."
End source

Reconstruction Original
  Claim C1 from LewisMatchScenario sentence 1
    Paraphrase "If the match had been struck, it would have lit."
    Formal meaning If it were the case that thisMatch is struck, then thisMatch
     is lit
  End claim
  Claim C2 from LewisMatchScenario sentence 2
    Paraphrase "If the match had lit, its flame would have been visible."
    Formal meaning If it were the case that thisMatch is lit, then thisMatch is
     visible
  End claim
  Claim Conclusion from LewisMatchScenario sentence 3
    Paraphrase "If the match had been struck, its flame would have been visible.
     "
    Formal meaning If it were the case that thisMatch is struck, then thisMatch
     is visible
  End claim

  Deduction CounterfactualConsequence
    Step VisibilityFollows
      From C1 and C2 conclude Conclusion
  End deduction
End reconstruction

Alternative DampMatch based on Original
  Change C1
    Original meaning If it were the case that thisMatch is struck, then
     thisMatch is lit
    Alternative meaning If it were the case that thisMatch is struck, then
     thisMatch is not lit
```

```
  End change

  Recheck the same deduction CounterfactualConsequence
End alternative
```

In this notebook the original deduction is rejected and Lean's model analysis certifies non-entailment: the selected damp situation satisfies the alternative assumptions but falsifies the visibility conclusion. The Counterfactual profile does not implement contradiction checking, so this example records no contradiction result. The dependency report in the Lean Infoview (not shown here) makes the reason visible at the language level. It names P1, P2, and P3 and the steps that use them. If an alternative changes a claim outside this dependency set, the recheck can remain accepted even when the alternative assumptions contain an independent contradiction. If so, the interface reports "This change does not affect this deduction", keeping the contradiction separate (see Section 7.2).

5.3 Diagnosing arguments from Lean's checked outcomes

Dialectic implements a series of diagnostic checks to give immediate feedback to the user. Syntax errors identify incomplete or malformed expressions; declaration and profile checks identify unknown vocabulary, broken source references, or meanings unavailable in the chosen logic. These checks support readability and native feedback (R1, R7), source traceability (R2), and controlled meaning (R3). Once the notebook can be interpreted, four outcomes are to be taken into account:

1. *Deduction acceptance*: do the written steps establish the target from their named premises? This makes inference visible (R4).
2. *Alternative recheck*: do those same steps still work after the declared change? Dependencies explain the effect of revision (R5).
3. *Inconsistency*: do the alternative assumptions yield a checked contradiction? This assesses the assumptions together (R6).
4. *Non-entailment*: is there a checked model satisfying the assumptions while falsifying the conclusion? This assesses what the assumptions support, independently of one attempted proof (R6).

All four are presented at the relevant notebook locations and in the Infoview (R7), sparing students from inspecting generated Lean proof terms.

Learning from inconsistency and non-entailment. A contradiction witness directs attention to assumptions that cannot jointly hold. The named claims give students a clear revision task: inspect their meanings and decide which commitment to reconsider, as Lean does not make that interpretive decision for them. An accepted deduction can coexist with this warning, because valid inference and jointly satisfiable premises are different questions. A countermodel – on the other hand – gives students a concrete situation to inspect: every premise holds, yet the conclusion fails. They can use it to identify a missing link or an overly strong conclusion. Failing to complete a deduction cannot provide this lesson with the same evidential force, since another deduction might succeed.

6 Logical Profiles

Logical profiles allow Dialectic to be a notebook interface without committing to a single logical framework for argumentation that stays equal for every notebook instance. Following common reasoning modes in philosophical argumentation [6,11], the system implements three logic profiles for reasoning in unary constructive logic, modal logic, and counterfactual reasoning.

ModalK. To use the ModalK profile a notebook must declare worlds, accessibility R, and valuations. The logical profile ModalK then instantiate propositions to be terms of type $World \to Prop$, with the standard K meanings [12]:

$$\begin{aligned} &\textit{in every possible world } P \mapsto \forall v,\ R(w,v) \to P(v),\\ &\textit{in some possible world } P \mapsto \exists v,\ R(w,v) \wedge P(v). \end{aligned}$$

No reflexivity, transitivity, symmetry, or Euclideanness is assumed. The Plantinga-inspired notebook[5] provides a minimal example [22]: necessarily, maximal greatness implies existence; necessarily, the candidate is maximally great; hence, necessarily, it exists. Replacing the second premise with possibility rejects the preserved step. Separately, Dialectic evaluates the declared finite model and submits counterexample evidence to Lean when the alternative assumptions hold and the target fails. Selecting this profile enriches the CNL with constructs that allow students to supply worlds, accessibility relations, and valuations. Model synthesis is not implemented in the current version of Dialectic.

Counterfactual. A counterfactual notebook distinguishes an actual situation from a declared set of relevant counterfactual situations. The phrase `If it were the case that P, then Q` means that every relevant P-situation satisfies Q. The implemented conditional at actual situation w is

$$\forall v,\ \textit{selected}(w,v) \to P(v) \to Q(v).$$

The consequence rule composes conditionals from P to Q and from Q to R only when both premises use the same declared set of relevant situations. The author supplies this set, and Dialectic checks composition against it. The fragment evaluates counterfactual truth over these situations: material implication and complete semantic accounts e.g., Lewis–Stalnaker [15,24], lie outside the scope of this first implementation. The match model declares an actual situation where the match is unstruck, unlit, and not visible, and a selected damp situation where it is struck but unlit and not visible. The declared `selected` predicate places the damp case in the set relevant to the actual situation. The alternative reconstruction in Code 1.8 exercises this semantics on the declared damp situation.

[5] See https://github.com/smnicoletti/dialectic/blob/main/Dialectic/Examples/ModalOntologicalArgument.lean

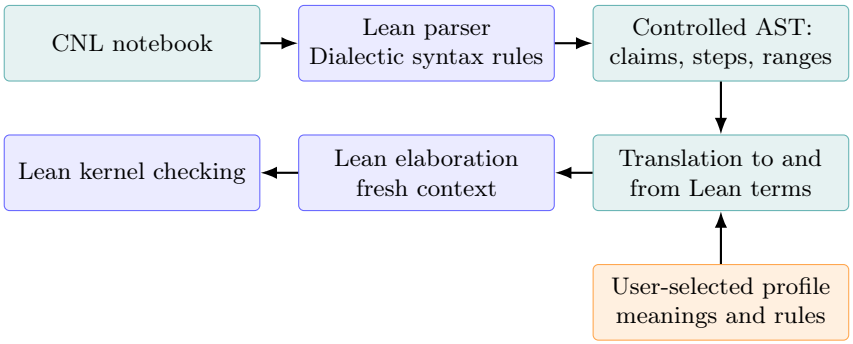

Fig. 1. From notebook to checked terms. Teal boxes identify Dialectic's contribution, orange its selected profile, and blue reused Lean infrastructure. Dialectic supplies the syntax rules consumed by Lean's parser. The controlled AST records the authored deduction, the selected profile then supplies its interpretation.

7 Implementation

Figure 1 summarizes the system implementation by describing the flow of information from a notebook to the Lean prover. As the input we provide a notebook and its logic profile. The notebook parser generates CNL syntax and validates the structure of names, vocabulary roles, source references, and reconstruction links. The selected profile then validates controlled meanings and deduction forms. Only after, the translation to Lean takes over and the resulting term can be checked. The complete open-source implementation is available in Dialectic's repository.[6]

7.1 Translation, elaboration, independent checking

The notebook can be parsed from within the Lean system by making use of Lean's extensible parser [19]. Once the controlled AST is built and validated from the input notebook the profile translates the accepted meaning into Lean propositions and the deduction steps into Lean term syntax.

Independence of reconstructions and elaboration Each notebook is logically independent and is parameterized by vocabulary, source, and logic profile that are shared between reconstruction, deduction, and alternatives. Additionally, a deduction is reinterpreted between the original and alternative interpretation. To allow this reinterpretation Dialectic provides a fresh context to each. This means that the underlying translations of original and alternative reconstruction are independent – they generate Lean terms that do not reference each other. This independence is assumed as an implementation choice in the artifact and not checked in any way by Lean or the logical profiles.

[6] https://github.com/smnicoletti/dialectic

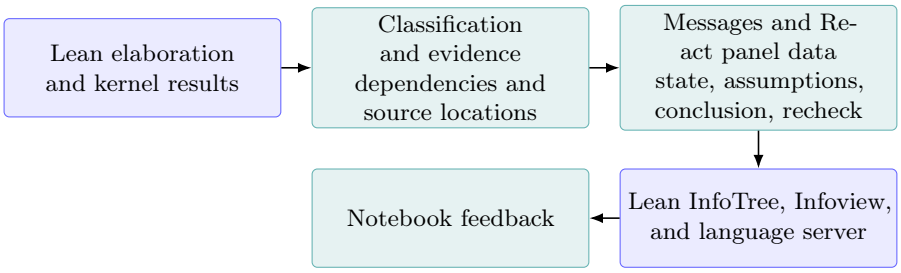

Fig. 2. From checking to understandable feedback. Teal denotes Dialectic; blue denotes Lean infrastructure. Applying a suggested edit starts a new forward checking cycle.

Untrusted implementation Dialectic is implemented as untrusted code, i.e., it cannot check theorems and all proofs have to be accepted by the Lean system. This is true also for contradictions and model evidence. Countermodels supplied through a logic profile are translated into evidence that Lean checks.

7.2 From checking to notebook feedback

Figure 2 shows how results and diagnostics flow from the Lean prover to the Dialectic notebook. Dialectic provides its own diagnostics in the same CNL the user writes and enriches Lean's Infoview window to render it.

Dialectic's diagnostics The Lean system provides interactive diagnostics in the Visual Studio Code (VSCode) editor by showing an additional panel named *Infoview*. This panel is part of the interactive experience of using the Lean system and provides information about the logical context and goals. It can be extended to display additional information which allows us to provide interactive diagnostics reusing Dialectic's CNL. We provide a structured presentation of the original and alternative reconstructions as well as the outcome of checking the deduction in both. Counter-examples are also shown in the diagnostics to explain that the assumptions hold while the conclusion does not. Additionally, we provide information about how the changes between the original and alternative reconstructions are meaningful, e.g. a changed assumption was not used in the deduction. Table 1 (page 15) distinguishes the available evidence from the limits of each implemented diagnostic procedure.

8 Discussion and Conclusion

Table 2 summarizes how the proposed artifact satisfies the requirements, providing implementation bits and notebooks that exercise them:

Table 1. Implemented negative-evidence checks and their limits.

Check	Checked evidence	Scope and limits
CoreLogic inconsistency	A selection of the premises from which we can derive `False`.	The procedure covers the four unary meaning forms used by the selected deduction. Failure to find a witness does not establish consistency.
CoreLogic non-entailment	A satisfying assignment of the premises that does not satisfy the conclusion.	Possible for finite models where the user supplies the finite domain and total assignment. Dialectic does not search unbounded domains, relations, or quantified models.
ModalK non-entailment	A satisfying assignment of the premises that does not satisfy a conclusion of the form $\Box\phi$.	The user supplies the worlds, accessibility relation, and valuation.
Counterfactual non-entailment	A declared relevant situation satisfies the antecedent and falsifies the target consequent.	Both premises use the same declared set of relevant situations. The fragment provides no closest-situation search or similarity ranking.

Table 2. How Dialectic addresses and exercises the requirements.

Requirement	Implementation and exercise
R1: Readable surface	Controlled sentences without Lean terms, as shown by the *Corroborated Testimony* notebook.
R2: Source traceability	Both notebooks separate source sentences, paraphrases, and formal meanings.
R3: Controlled meaning	Logical profiles provide semantics mapping for CNL terms.
R4: Visible inference	Named premises, results, and intermediate claims, e.g., in the *Corroborated Testimony* deduction.
R5: Explicit alternatives	Alternative meanings and fresh deduction checks in both presented notebooks.
R6: Negative evidence	`CounterfactualMatch.lean` reports deduction rejection and non-entailment separately.
R7: VS Code and Lean embedding	Lean notebooks expose deduction state, diagnostics, and Quick Fixes through the Infoview.

Limitations Dialectic currently supports a small number of logical fragments. CoreLogic provides unary constructive reasoning but not binary relations or classical first-order logic. Countermodel support checks user-supplied finite models and does not perform general model search. Translation faithfulness rests on the profile author: Lean checks the generated evidence, but no mechanized proof connects a profile's translation to its intended semantics. Furthermore, the educational value of the environment has not yet been evaluated with students.

Future work We plan to add a classical first-order profile with binary predicates, identity, and richer quantifier scope. Bounded countermodel search, informed by Nitpick [2], could provide feedback without requiring students to supply models. Further work will mechanize translation correctness, assess the environment with students, and decouple the notebook interface from its logic profiles so that profiles can supply their own CNL fragments.

Generative-AI usage disclosure. Generative-AI tools assisted with development and manuscript preparation. The authors retain responsibility for all materials and claims.

References

1. Barras, B., Boutin, S., Cornes, C., Courant, J., Coscoy, Y., Delahaye, D., de Rauglaudre, D., Filliâtre, J.C., Giménez, E., Herbelin, H., et al.: The coq proof assistant reference manual. INRIA, version **6**(11), 17–21 (1999)
2. Blanchette, J.C., Nipkow, T.: Nitpick: A counterexample generator for higher-order logic based on a relational model finder. In: Kaufmann, M., Paulson, L.C. (eds.) Interactive Theorem Proving. pp. 131–146. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
3. Bobonich, C.: How to reconstruct an argument. Stanford University teaching materials, https://web.stanford.edu/~bobonich/argument%20reconstruction/arg.recon.main.html, accessed 18 September 2026
4. Dutilh Novaes, C.: Argument and argumentation. In: Zalta, E.N., Nodelman, U. (eds.) The Stanford Encyclopedia of Philosophy. Metaphysics Research Lab, Stanford University, fall 2022 edn. (2022), https://plato.stanford.edu/archives/fall2022/entries/argument/
5. Elkind, L.D.C.: Computer verification for historians of philosophy. Synthese **200**(3), 1–28 (2022). https://doi.org/10.1007/s11229-022-03678-y, article 242
6. Ferrario, R.: Counterfactual reasoning. In: International and Interdisciplinary Conference on Modeling and Using Context. pp. 170–183. Springer (2001)
7. Fuenmayor, D., Benzmüller, C.: Computer-assisted reconstruction and assessment of E. J. Lowe's modal ontological argument. Archive of Formal Proofs pp. 1–19 (2017), https://www.isa-afp.org/entries/Lowe_Ontological_Argument.html
8. Fuenmayor, D., Benzmüller, C.: A case study on computational hermeneutics: E. J. Lowe's modal ontological argument. IfCoLog Journal of Logics and Their Applications **5**(7), 1567–1603 (2018), https://hdl.handle.net/10993/37444
9. Fuenmayor, D., Benzmüller, C.: Computational hermeneutics: An integrated approach for the logical analysis of natural-language arguments. In: Liao, B., Ågotnes, T., Wáng, Y.N. (eds.) Dynamics, Uncertainty and Reasoning: The Second Chinese Conference on Logic and Argumentation, pp. 187–207. Springer, Singapore (2019). https://doi.org/10.1007/978-981-13-7791-4_9, https://doi.org/10.1007/978-981-13-7791-4_9
10. Glick, E.: 24.241 Logic I. MIT OpenCourseWare, Massachusetts Institute of Technology (2009), https://ocw.mit.edu/courses/24-241-logic-i-fall-2009/, fall 2009 course materials; accessed 18 September 2026
11. Johnson-Laird, P.N., Ragni, M.: Reasoning about possibilities: Modal logics, possible worlds, and mental models. Psychonomic Bulletin & Review **32**(1), 52–79 (2025)
12. Kripke, S.A.: Semantical analysis of modal logic i: Normal modal propositional calculi. Zeitschrift für mathematische Logik und Grundlagen der Mathematik **9**(5–6), 67–96 (1963). https://doi.org/10.1002/malq.19630090502
13. Kuhn, T.: How to evaluate controlled natural languages. In: Pre-Proceedings of the Workshop on Controlled Natural Language (CNL 2009) (2009). https://doi.org/10.5167/uzh-23864, https://arxiv.org/abs/0907.1251
14. Kuhn, T.: A survey and classification of controlled natural languages. Computational Linguistics **40**(1), 121–170 (2014). https://doi.org/10.1162/COLI_a_00168
15. Lewis, D.K.: Counterfactuals. Blackwell and Harvard University Press, Oxford and Cambridge, MA (1973)
16. Massot, P.: Teaching mathematics using Lean and controlled natural language. In: 15th International Conference on Interactive Theorem Proving (ITP 2024). Leibniz International Proceedings in Informatics (LIPIcs), vol. 309, pp. 27:1–27:19. Schloss

Dagstuhl – Leibniz-Zentrum für Informatik (2024). https://doi.org/10.4230/LIPIcs.ITP.2024.27
17. Michels, R.: The formalization of arguments: An overview. Dialectica **74**(2), 179–213 (2020). https://doi.org/10.48106/dial.v74.i2.01
18. de Moura, L., Kong, S., Avigad, J., van Doorn, F., von Raumer, J.: The lean theorem prover (system description). In: Felty, A.P., Middeldorp, A. (eds.) Automated Deduction - CADE-25. pp. 378–388. Springer International Publishing, Cham (2015)
19. Moura, L.d., Ullrich, S.: The lean 4 theorem prover and programming language. In: Platzer, A., Sutcliffe, G. (eds.) Automated Deduction – CADE 28. pp. 625–635. Springer International Publishing, Cham (2021)
20. Murphy, L., Yang, K., Sun, J., Li, Z., Anandkumar, A., Si, X.: Autoformalizing Euclidean geometry. In: Proceedings of the 41st International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 235, pp. 36847–36893. PMLR (2024), https://proceedings.mlr.press/v235/murphy24a.html
21. Paulson, L.C.: Isabelle: The next 700 theorem provers. In: Logic and computer science. vol. 31, pp. 361–386 (1990)
22. Plantinga, A.: The Nature of Necessity. Clarendon Press, Oxford (1974). https://doi.org/10.1093/0198244142.001.0001, chapter 10
23. Rushby, J.: Mechanized analysis of Anselm's modal ontological argument. International Journal for Philosophy of Religion **89**(2), 135–152 (2021). https://doi.org/10.1007/s11153-020-09768-6
24. Stalnaker, R.C.: A theory of conditionals. In: Rescher, N. (ed.) Studies in Logical Theory, pp. 98–112. Basil Blackwell, Oxford (1968)
25. Tran Minh, F., Gonnord, L., Narboux, J.: A Lean-based language for teaching proof in high school. In: Conference on Intelligent Computer Mathematics 2025. pp. 447–467. Springer (2025). https://doi.org/10.1007/978-3-032-07021-0_25
26. University of Oxford, Faculty of Philosophy: Course descriptions FPE: Introduction to logic. Official course descriptions, https://www.philosophy.ox.ac.uk/node/82976, accessed 18 September 2026
27. Voigt, C.: Argdown and the stacked masonry layout: Two user interfaces for non-expert users. In: Computational Models of Argument (COMMA 2014). pp. 483–484. IOS Press (2014). https://doi.org/10.3233/978-1-61499-436-7-483
28. Wu, Y., Jiang, A.Q., Li, W., Rabe, M., Staats, C., Jamnik, M., Szegedy, C.: Autoformalization with large language models. In: Advances in Neural Information Processing Systems. vol. 35, pp. 32353–32368 (2022). https://doi.org/10.52202/068431-2344
29. Yablo, S.: 24.244 Modal Logic: Syllabus. MIT OpenCourseWare, Massachusetts Institute of Technology (2015), https://ocw.mit.edu/courses/24-244-modal-logic-spring-2015/pages/syllabus/, spring 2015 course materials; accessed 18 September 2026