

Do Argumentation Schemes Help? An Assessment of Their Role in Formalising Natural Language Arguments

Ameer SAADAT-YAZDI^{a,1}, Victor DAVID^a, Anthony HUNTER^b and
Serena VILLATA^a

^a *Université Côte d’Azur, Inria, CNRS, I3S*

^b *University College London*

Abstract. Argument schemes are structured templates that capture common patterns of reasoning. We explore whether schemes can guide the automatic construction of formal proofs that derive an argument’s claim from its premises. To do this, we propose a pipeline that translates natural language into proofs in Isabelle/HOL, using argument schemes. The schemes are formalised systematically as inference rules in higher-order logic, and are used to structure proof generation within an iterative LLM-driven reasoning loop. Evaluating on a dataset of arguments that fully instantiate their schemes across 18 scheme types, we compare this scheme-guided approach against a baseline in which no scheme is provided. There are two sources of incompatibility between schemes and natural language arguments: one where the scheme’s conclusion does not directly match the argument’s claim, and one where the natural language premises are insufficient to satisfy the scheme’s requirements. We find that scheme guidance only marginally improves results overall, though when the scheme’s requirements are satisfied, the pipeline reliably closes the final inferential step; the bottleneck is that natural language arguments rarely make the required information explicit, and LLMs struggle to supply the necessary connections. Our error analysis reveals four key factors contributing to this incompatibility, and we suggest future directions to address them.

Keywords. argument scheme, semantic parsing, neurosymbolic, large language model, theorem proving

1. Introduction

Argument schemes play a central role in analysing argumentation. Walton defines them as forms of argument that capture common patterns of reasoning in everyday and specialised discourse [1]. Identifying an argument’s scheme enables evaluation of inferential strength [2] and specifies the premises required to justify a claim. Crucially, schemes are conceived as explanations of the underlying reasoning mechanism: they make explicit the inferential step that licenses the move from premises to claim.

For example, consider an instantiation of the expert opinion scheme (Table 1) in Example 1. The scheme does not merely categorise the argument; it specifies the mech-

¹Corresponding Author: Ameer Saadat-Yazdi, ameer.saadaty-yazdi@inria.fr

anism by which the claim is warranted, given that the source is a domain expert whose assertion carries epistemic authority. A complete formalisation must therefore satisfy these additional constraints: Dr. Smith must be an expert in the relevant domain, and his assertion must imply the claim.

Example 1 — Argument From Expert Opinion

“Dr. Smith, a palliative care physician, says euthanasia is the most humane option for the terminally ill. Palliative care physicians specialise in end-of-life treatment. Therefore, euthanasia should be permitted for the terminally ill.”

Table 1. Argument scheme for argument from expert opinion.

Component	Description
Expertise Premise	Source E is an expert in domain D
Assertion Premise	E asserts that proposition A (in domain D) is true
Conclusion	A is true

Work on argument schemes in computational settings has followed two main directions. In formal reasoning, schemes are encoded as inference rules within structured knowledge bases, often supporting non-monotonic reasoning and explanation generation [3,4,5]. In contrast, research in computational linguistics treats schemes as analytical tools for identifying argument patterns [6,7,8]. These traditions have remained largely separate, yet could benefit from each other. Argument mining could populate symbolic knowledge bases, while symbolic argumentation provides explainability and robustness to neural natural language processing systems. The main challenge between the two fields is that natural language arguments are often incomplete, while symbolic reasoning systems require all assumptions and facts to be made explicit. Recent work has begun to bridge this gap through *enthymeme decoding*. In particular, [9,10] propose logic-based frameworks that study how to select appropriate reconstructions of implicit premises and inferential steps. However, how to obtain such reconstructions from natural language remains largely unexplored. As an initial step, this paper addresses this issue by formalising arguments into logic while using argument schemes to guide the generation of the missing information required to license the inference.

A central challenge is that schemes seldom align directly with surface text, and their use in formal settings often relies on manual engineering or brittle rules. If schemes genuinely capture reasoning mechanisms, they should guide the construction of formal proofs from natural language premises, a possibility not yet fully explored.

Recent neurosymbolic work shows that the gap between natural language and formal reasoning is narrowing. Quan et al. [11] propose an iterative framework in which an LLM constructs proof sketches refined through theorem-prover feedback, providing a methodological foundation for our approach; however, their method relies on large proprietary models to recover implicit reasoning. In contrast, we aim to leverage smaller, locally deployable models, using argument schemes to guide the reconstruction of missing information. While such approaches ground natural language arguments in formal systems, they treat argument structure as flat, connecting premises to claims without exploiting underlying inferential templates. The contribution of scheme-guidance therefore remains an open question.

This paper addresses that question. Given arguments that are complete instantiations of known schemes, we ask: *Does providing the argument scheme as explicit scaffolding improve proof construction?* We propose a pipeline that translates natural language premises and claims into logical statements and attempts to derive the claim through valid inferential steps. This formalisation is carried out in the Isabelle/HOL [12] theorem-proving environment, using schemes within an iterative loop where an LLM proposes and refines proofs while the prover verifies them.

By comparing this scheme-guided process with a baseline without schemes, we quantify its contribution. We further analyse where and why schemes diverge from natural language, showing that many failures correspond to missing information required for inference, and outlining directions for improving their integration with computational models of reasoning.²

2. Methodology

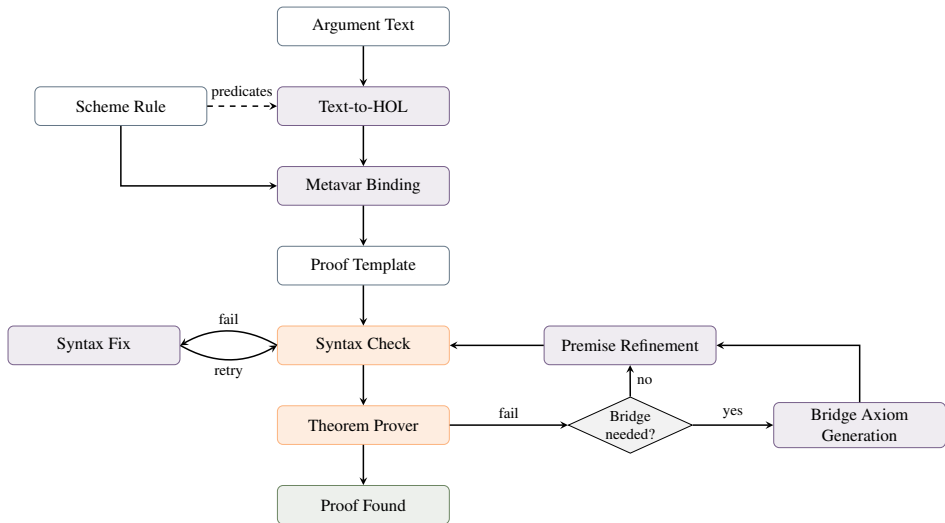


Figure 1. Overview of the proof construction pipeline (scheme version). LLM steps (purple) translate natural language to formal representations, instantiate scheme variables, and generate candidate axioms. Isabelle steps (orange) verify syntactic correctness and attempt automated proof. On failure, the system iteratively refines the theory via bridging axioms or axiom revision until success or a maximum iteration limit. The baseline is the same except that the Scheme Rule, Metavar Binding and Proof Template boxes are omitted.

Our approach uses argument schemes to bridge natural language and formal reasoning, translating textual arguments into Isabelle/HOL proofs in three stages: formalising components, instantiating a scheme-based proof template, and iteratively refining the proof using theorem-prover feedback. We use Isabelle/HOL because higher-order logic is required to represent schemes, in particular those involving sources or properties, and

²The code used is available at: https://github.com/ameertg/formalising_arg_schemes

because Sledgehammer can automatically discharge inference steps, avoiding low-level proof generation. Figure 1 illustrates the pipeline.

2.1. Text-to-HOL

Each sentence in the argument is mapped to a typed higher-order predicate calculus; we use higher-order because several scheme predicates take propositions as arguments, as in $\text{Asserts}(E, A)$ where A is a formula rather than a term. We follow Quan et al., where sentences are first syntactically parsed to identify their predicate-argument structure. Each sentence is then mapped to a typed predicate calculus with four base types: agent, domain, and polymorphic types 'a and 'b for miscellaneous entities.

The translation proceeds incrementally: the claim is processed first, followed by the premises one at a time. In each step, the LLM is provided with a dictionary of previously used predicates and constants, their types and natural language explanations of their meaning. To ensure consistency and prevent predicate duplication, the LLM is prompted to use existing predicates and generate new ones only if necessary. The resulting formalisation of Example 1 is shown directly in Example 2 below.

2.2. Formalising Schemes

Argument schemes are not natively formal objects; they are typically specified in structured natural language. We treat their formalisation as a knowledge representation problem: how to encode schemes as general, reusable inference rules while preserving their intended semantics. Each scheme is represented as a typed, meta-level inference rule in which predicates capture core relational structure (e.g. expertise, assertion, domain relevance) and metavariables denote the entities involved. While we aim to remain faithful to Walton's original definitions [1], existing formalisation efforts either leave individual representations to the practitioner [13] or are restricted in scope [14], and no systematic account exists of how schemes should be represented in general. This is a non-trivial balancing act: diverging too far from Walton's original definition risks misalignment with the natural language arguments that the formalisation pipeline must parse, while adhering too closely preserves ambiguities in Walton's informal presentations that formal logic cannot tolerate.

To ensure consistency, we impose two design constraints. First, all variables must be *relationally grounded*: each must appear in at least one predicate connecting it to other variables and be referenced by more than one premise, preventing unbound or context-dependent entities. Second, conjunctive premises are decomposed into atomic premises so that each inferential step can be independently verified. This decomposition is also generally useful for finer-grained analyses of similarity, minimality, and relations [15].

These constraints lead to slight but principled deviations from Walton's informal presentations. For example, in the expert opinion scheme, the relevance of the proposition to the expert's domain is sometimes implicit or embedded within other premises. In our formalisation, this information is made explicit as a separate premise, ensuring that all required inferential links are available for formal verification compared with Table 1:

```
axiomatization where
expert_opinion:
  "[ Expert E D; PropositionInDomain A D; Asserts E A ] ==> A"
```

We choose to enforce a shared predicate vocabulary between schemes and the semantic parse of natural language arguments, ensuring that predicates appearing in the argument are not defined or reinterpreted multiple times. Typed scheme predicates are introduced before sentence formalisation, ensuring that the generated predicates align with the scheme vocabulary and share a common ontology.

2.3. Creating Proof Templates

This step most distinguishes our approach from that of Quan et al.: rather than asking the LLM to generate a proof from scratch, we provide a structured template that encodes the scheme's inferential structure directly.

At this stage, the outputs of the previous steps are combined to produce an initial proof. In Isabelle, a theorem takes the form *assumes P, shows Q*: given a named assumption *asm* (the formalised natural language premise), derive a claim. The LLM binds the scheme's metavariables to concrete terms derived from the formalised argument, instantiating the template for the specific claim at hand. Each scheme premise (such as *Expert(E,D)* in the expert opinion scheme above) becomes a *have* step: a named intermediate sub-goal. Initially, every *have* step carries both a *using asm* clause (indicating it may rely on the theorem's assumption) and a *sorry* placeholder, Isabelle's built-in mechanism for deferring a proof obligation. The critique loop that follows then replaces each *sorry* with a verified tactic. Finally, the *show* step assembles the intermediate sub-goals into the claim by instantiating the scheme rule via Isabelle's `[OF ...]` combinator. The proof template for Example 1 after binding the expert opinion scheme's metavariables is shown in Example 2.

Example 2 — Proof Template for Expert Opinion

```
theorem ExpertOpinion:
  (* HOL translated premises *)
  assumes asm_1: "Asserts DoctorSmith (HumaneOption Euthanasia TerminallyIll)
    & PalliativeCarePhysician DoctorSmith"
    and asm_2: "\<forall> x. PalliativeCarePhysician x -->
    Expert x EndOfLifeTreatment"
  (* HOL translated claim *)
  shows "ShouldBePermitted Euthanasia TerminallyIll"

proof -
  have step_1: "Expert DoctorSmith EndOfLifeTreatment"
  sorry
  (* Scheme gap *)
  have step_2: "PropositionInDomain (HumaneOption Euthanasia TerminallyIll)
    EndOfLifeTreatment"
  sorry
  have step_3: "Asserts DoctorSmith (HumaneOption Euthanasia TerminallyIll)"
  sorry
  (* Claim gap *)
  show ?thesis
  using expert_opinion[OF step_1 step_2 step_3]
  sorry
qed
```

2.4. Iterative Proof Completion

Each `sorry` is replaced with a `sledgehammer` call. `Sledgehammer`³ is Isabelle’s built-in tool that invokes external automated theorem provers to discharge goals within the current proof context by applying deductive inference rules to the available assumptions; when successful, proofs are reconstructed and inserted into the proof script.

When proof steps remain unresolved, a sequential decision procedure determines how to proceed: if unsolved scheme premises remain, additional premises are generated to connect existing premises to the scheme’s requirements, this is the *scheme gap*. Using a symbolic solver, we check for premises that are logically equivalent to, or merely restate, the claim. These are rejected to avoid circular reasoning. If all scheme premises are satisfied but the final step fails, we have a *claim gap* and a bridging premise is generated to close the gap between the scheme’s conclusion and the original argument’s claim. The updated proof is rechecked for syntax, corrected if needed, and retried until success or a maximum number of iterations is reached.

In Example 2, `sledgehammer` will likely close `step_1` and `step_3` in the first iteration. However, both gaps still arise: `step_2` is a scheme gap: `PropositionInDomain` is never stated in the text, so a premise asserting that the statement `HumaneOption Euthanasia TerminallyIll` falls within the domain of `EndOfLifeTreatment` must be generated. The `show` step has a bridge gap: the scheme concludes `HumaneOption ...` while the argument concludes `ShouldBePermitted ...` requiring a generated premise connecting the two before the proof can close.

3. Experiments and Results

To evaluate whether argument schemes aid formal proof construction, we use arguments that fully instantiate their schemes, ensuring that failures reflect mismatches between natural language and schemes rather than missing premises. As a baseline, we adapt Quan et al. [11], generating proofs without scheme guidance by omitting scheme predicates, metavariable binding, and proof-template generation from Figure 1.

3.1. Dataset

We use the NLAS dataset [16], which consists of 1893 English and 1917 Spanish arguments synthetically generated directly from Walton’s scheme definitions [1] and manually verified for plausibility. To our knowledge, NLAS is the only available dataset providing complete, verified instantiations of a broad range of Walton’s schemes with no missing premises, making it uniquely suited to our purposes: any failures can be attributed to the mismatch between natural language and formal structure rather than to incomplete arguments. The arguments are labelled with their topic, scheme, and stance (pro or con).

Of the 20 schemes in NLAS, we exclude *ad hominem* and example arguments as we were not able to formalise these as deductive inference rules in Isabelle, leaving the 18 schemes shown in Table 2. We select a stratified random subset of 200 English-language arguments, with each scheme represented by at least 10 examples.

³<https://isabelle.in.tum.de/dist/doc/sledgehammer.pdf>

3.2. Setup

We evaluate 3 comparably sized open-source HuggingFace models for reproducibility: Qwen (v2.5)⁴, Codestral⁵, and Gemma (v4)⁶. These models are all trained on coding and reasoning tasks, making them well-suited for generating syntactically correct formal logic. They also represent the state of the art among three different open-source model families with mid-sized variants available. All models were 8-bit quantised.

We set a maximum of 5 iterations per stage (binding, syntax, and premise generation), resulting in up to $5 + 5 \times 5 = 30$ LLM and Isabelle calls per example. Examples still syntactically incorrect after 5 iterations are terminated early and marked as failures.

3.3. Results

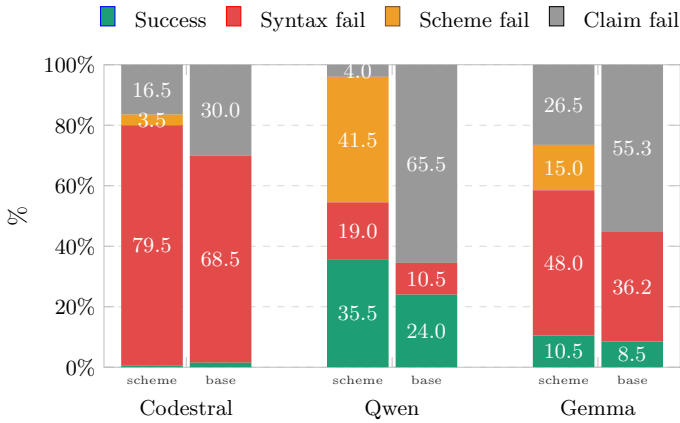


Figure 2. Performance of scheme vs baseline to proof construction. Scheme failure denotes unsatisfied scheme premises, while claim failure occurs when all intermediate steps are solved but the claim is not proven.

Figure 2 summarises performance across the three models and two pipelines. We report four outcome categories: success, syntax failure (the Isabelle theory is rejected before any proof search), scheme failure (the scheme premises cannot be satisfied), and claim failure (all scheme premises are closed but the claim remains open). Note that, syntax failure implies scheme and claim failure, and scheme failure implies claim failure. To avoid double-counting, each example is assigned to its earliest failure. In the baseline, all logical failures are recorded as claim failures since no scheme is provided.

Model capability dominates. The most striking pattern is the performance gap between Qwen and the other two models. While Qwen achieves 35.5% success under scheme guidance and 24.0% in the baseline, the other models fail almost entirely at the syntactic level, preventing scheme guidance from taking effect. This suggests the task is gated by a minimum threshold of code generation capability: models below it cannot participate meaningfully regardless of scheme guidance.

⁴<https://huggingface.co/Qwen/Qwen2.5-Coder-32B-Instruct>

⁵<https://huggingface.co/mistralai/codestral-22B-v0.1>

⁶<https://huggingface.co/google/gemma-4-31B-it>

Syntactic failures decomposed. Codestral fails syntactically in 79.5% of scheme-guided cases and 68.5% of baseline cases, likely due to its smaller parameter count relative to the other models. 76% of Gemma’s syntactic failures are due to structurally malformed Isabelle, with the remaining being type errors. Qwen’s syntax failures skew toward type conflicts (47%), where the same variable is applied to predicates requiring incompatible types. Outright malformed syntax occurs in only 34% of cases, indicating that Qwen has internalised Isabelle’s surface grammar but struggles to create consistent formulae that respect the type system. Section 4.1 explores these kinds of errors in more detail. Moving forward, this could be improved by taking advantage of larger models, which are more capable of representing lesser-known languages like Isabelle.

Effect of scheme guidance. For Qwen, scheme guidance improves success by 11.5 percentage points; however 41.5% of scheme-guided attempts fail at filling the scheme failure gap. Of successfully instantiated schemes only 10.0% suffer from claim failure, confirming that when a scheme is successfully instantiated, it reliably closes the final inferential step. The bottleneck, discussed in Section 4.2, is deriving the scheme premises from the text in the first place.

4. Analysis

In this section, we analyse Qwen’s outputs to identify systematic error patterns. Since Codestral and Gemma fail predominantly at the syntactic level, analysing their outputs yields little insight into the deeper incompatibilities between schemes and natural language; we therefore focus on Qwen, whose outputs reach the scheme and claim stages often enough to reveal systematic patterns. The examples given are all taken directly from the dataset and the output of the model.

4.1. Syntactic Failure

Syntactic failure arises when the formalised theory is rejected by Isabelle before any proof search can begin. Two distinct causes are observed.

Type conflict. The formaliser assigns a single quantified variable to predicates that inhabit different semantic types. For example, when two predicates are assigned to different types in the formalisation (e.g. ‘Medicine’ inhabits `domain` while ‘Patient’ inhabits `agent`), yet the formalised claim applies them to the same variable x . Isabelle cannot unify the two concrete types for x and rejects the theory outright.

LLM misconstruction. The LLM faces challenges rooted in the logical content of generated premises. The model frequently misapplies logical connectives to typed constants rather than propositions, or applies predicates to arguments of the wrong arity. A greater difficulty arises with higher-order schemes, for details see Section 4.3.

4.2. Failed Scheme Premises

Scheme failure occurs when an argument premise is weaker than the corresponding scheme requirement, and the system fails to generate the commonsense bridge needed to

Table 2. Success rate of (S)cheme vs (B)aseline approaches by argumentation scheme (Qwen2.5-Coder-32B).

Scheme	S	B	Scheme	S	B
Best Explanation	63.6	18.2	Popular Practice	36.4	27.3
Analogy	54.5	54.5	Sunk Costs	27.3	27.3
Popular Opinion	54.5	0.0	Ignorance	27.3	45.5
Sign	54.5	27.3	Waste	18.2	18.2
Established Rule	50.0	33.3	Expert Opinion	18.2	18.2
Witness Testimony	50.0	25.0	Threat	18.2	18.2
Position To Know	45.5	0.0	Slippery Slope	18.2	0.0
Verbal Classification	45.5	18.2	Cause To Effect	9.1	36.4
Inconsist. commitment	36.4	18.2	Precedent	9.1	45.5

strengthen it. The sunk costs scheme of Example 3, for instance, requires the premise “I am precommitted to euthanasia as an end of life option” to satisfy the sunk cost scheme.

When the system instead sees a weak existential claim: that at some point the agent has invested time and resources, the required predicate cannot be derived, and the goal remains unsolved across all refinement iterations. The LLM fails to make a connection between the second premise and the speaker being precommitted.

Example 3 — Scheme failure (sunk costs)

Premises:

1. There is a choice at the end-of-life between euthanasia and not choosing euthanasia.

2. At some point in my life, I have likely invested significant time, energy, and resources into creating an end-of-life plan that includes euthanasia as a viable option.

Claim: I should choose euthanasia as my end-of-life option because I have already made a significant investment in considering it and planning for it.

4.3. Performance Shift Across Schemes

Table 2 shows a marked inversion between the scheme-guided and baseline conditions. Schemes with a simple propositional structure benefit most from explicit instantiation: best explanation has 63.6% success with schemes versus 18.2% without, while popular opinion and position to know similarly drop to 0% in the baseline. This indicates that, for these schemes, the inferential structure is not recoverable from natural language alone and must be explicitly supplied.

In contrast, precedent degrades from 45.5% without schemes to 9.1% with them. To see why, consider what the scheme requires in Table 3. In our type system, F and G are predicates of type $'a \Rightarrow bool$, so the established rule is a higher-order predicate, $EstablishedRule\ F\ G$. As this argument models an exception, we have to use the higher-order relationship to avoid conflict with the minor premise, since otherwise

$$(\forall x.F(x) \rightarrow G(x)) \wedge F(a) \wedge \neg G(a)$$

would lead to an inconsistency.

The model consistently treats these higher-order slots as first-order, passing individuals or propositions where predicates/functions are required. This reflects a representational mismatch between the natural language framing, which speaks of rules and prop-

Table 3. [Refutation] Scheme for Argument from Precedent

Component	Description
Major premise	Generally, according to the established rule, if x has property F , then x also has property G .
Minor premise	In this legitimate case, a has property F but does not have property G .
Conclusion	Therefore, an exception to the rule must be recognised, and the rule must be appropriately modified or qualified.

erties being modified or recognised, and the higher-order vocabulary needed to express those relationships formally. This is a gap that the refinement loop cannot easily close since Isabelle’s type unification errors do not indicate what kind of term is expected and because the LLM has limited exposure to higher-order logics.

4.4. Stance Asymmetry

Looking at the argument stance, pro arguments significantly outperform con arguments under scheme-guided construction ($p < 0.004$, two-proportion z-test), while no significant difference is observed in the baseline without schemes. This suggests a structural asymmetry: argumentation schemes are canonically formulated in their positive form. Constructing con arguments requires negating or inverting the claim, disrupting alignment with the scheme’s premises and making reconciliation with the scheme unreliable.

Example 4 illustrates this pattern for the verbal classification scheme, where the required negation of the classification predicate cannot be derived from the scheme as canonically defined. When the target claim is negative: $\neg \text{ClassifiedAs}(x, \text{ResponsibleCapability})$, the system has no way of deriving a negation of the claim from the scheme. Instead, the system introduces the premise $\text{ClassifiedAs}(x, \neg \text{ResponsibleCapability})$, which does not entail the required claim. The negative claim, therefore, cannot be recovered from the scheme’s output, and the final proof step fails.

Example 4 — Stance asymmetry (verbal classification, con)

Topic: Internet access for children **Stance:** con

Premises:

- Children under 12 lack the cognitive ability to discern appropriate from inappropriate online content.
- Individuals lacking this ability cannot be classified as having the responsible capability to access the internet without supervision.

Claim: Children under 12 cannot be classified as having the responsible capability to access the internet without supervision.

Formalised Claim:

$$\forall x. \text{Child}(x) \wedge \text{Under12}(x) \rightarrow \neg \text{ClassifiedAs}(x, \text{ResponsibleCapability})$$

Scheme:

$$\frac{\text{HasProperty}(x, f) \quad \forall y. \text{HasProperty}(y, f) \rightarrow \text{ClassifiedAs}(y, g)}{\text{ClassifiedAs}(x, g)}$$

4.5. Nested Reasoning in Single-Scheme Arguments

Scheme-Conclusion failures exhibit a consistent pattern: the scheme licenses a specific conclusion while the argument advances a related but structurally richer claim involving generalisation or quantification. Establishing the connection requires additional reason-

ing not supported by the available premises, so the final inferential step fails despite all prior scheme-related steps succeeding.

One cause of this failure is that many arguments labelled with a single scheme in fact embed an additional inference within the claim. Linguistic markers such as “in order to”, “so that”, or “such that” often signal that the claim is goal-directed rather than atomic, implicitly combining a deontic claim with a further consequentialist step that falls outside the scheme’s scope. When formalising the natural language under the assumption that the claim is an atomic argumentative unit, these markers are converted to implications. When that consequentialist component is not independently supported, the overall inference fails even if the scheme itself is correctly instantiated.

Example 5 shows this: the waste scheme closes, but the claim’s “in order to” and “and that” clauses introduce an implication requiring an unsupported additional step.

Example 5 — Nested reasoning (waste)

Premises:

1. If children are denied internet access, all the previous efforts made towards incorporating technology into education will be wasted.

2. If those efforts are wasted, children will be ill-prepared for the increasingly technology-dependent future job market.

Claim: Children ought to be provided internet access *to ensure that* the previous efforts towards incorporating technology in education are not wasted *and that* they are equipped for the future job market.

5. Conclusion

Returning to our research question as to whether schemes aid in constructing formal proofs of arguments, our findings suggest that schemes offer only a marginal benefit on a case-by-case basis. Schemes are too restrictive to fully capture the structure of natural language reasoning, even under the most favourable conditions: synthetic arguments generated directly from scheme definitions. This can be linked to four key factors shown in Table 4.

Table 4. Key factors contributing to the formalisation gap.

Structural mismatch	Arguments fail to fully instantiate the scheme’s reasoning structure
Type complexity	Language models struggle with complex type constraints
Commonsense reasoning	LLMs cannot consistently produce the right commonsense rules
Inference compression	Arguments merge multiple inferences in a single claim

Several directions follow naturally from this work. First, the generated premises required to complete proofs are themselves implicit premises, making this pipeline a natural basis for enthymeme reconstruction in natural language arguments. Second, schemes as currently used are binary: when an argument imperfectly instantiates a scheme, the scheme cannot compensate for the mismatch and becomes an obstacle rather than a guide. A more flexible approach would treat schemes as soft scaffolding, informing and constraining proof construction where they apply cleanly, but allowing partial use, adaptation, or graceful abandonment when the argument resists full instantiation. Third, argument schemes as they stand lack a proper formalisation: future work should investigate how higher-order reasoning or meta-logical frameworks can be used to formalise them

more rigorously, and how negation and refutation patterns can be systematically handled within scheme-based inference. Finally, this work assumes the argument's scheme is known in advance. Relaxing this assumption by first selecting a subset of candidate schemes (e.g., via classification), and then validating them through formalisation and proof construction, would bring the approach closer to realistic settings. In practice, an argument may instantiate multiple schemes depending on its interpretation or complexity, making joint scheme identification and validation essential.

Acknowledgements

All authors are supported by the French National Research Agency (ANR grant AGGREEY ANR-22-CE23-0005). The work by Victor David was supported by the French government, managed by the Agence Nationale de la Recherche under the Plan d'Investissement France 2030, as part of the Initiative d'Excellence d'Université Côte d'Azur under the reference ANR-15-IDEX-01. Experiments presented in this paper were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see <https://www.grid5000.fr>).

References

- [1] Walton D, Reed C, Macagno F. *Argumentation Schemes*. Cambridge University Press; 2008.
- [2] Walton D, Godden D. The Nature and Status of Critical Questions in Argumentation Schemes. *OSSA Conference Archive*. 2005 Jun.
- [3] Kökciyan N, Parsons S, Sassoon I, Sklar E, Modgil S. An Argumentation-Based Approach to Generate Domain-Specific Explanations. *Multi-Agent Systems and Agreement Technologies*. 2020 Apr.
- [4] Atkinson K, Bench-Capon T, Modgil S. Argumentation for Decision Support. In: Bressan S, Küng J, Wagner R, editors. *Database and Expert Systems Applications*. Springer; 2006. .
- [5] Vassiliades A, Bassiliades N, Patkos T. Argumentation and explainable artificial intelligence: a survey. *The Knowledge Engineering Review*. 2021 Jan;36.
- [6] Saadat-Yazdi A, Kökciyan N. Beyond Recognising Entailment: Formalising Natural Language Inference from an Argumentative Perspective. In: Ku LW, Martins A, Srikumar V, editors. *Proceedings of the 62nd Annual Meeting of the ACL (Volume 1: Long Papers)*. Bangkok, Thailand: ACL; 2024. .
- [7] Heinrich M, Khatib KA, Stein B. Multi-Class versus Means-End: Assessing Classification Approaches for Argument Patterns. In: Chistova E, Cimiano P, Haddadan S, Lapesa G, Ruiz-Dolz R, editors. *Proceedings of the 12th Argument mining Workshop*. Vienna, Austria: ACL; 2025. .
- [8] Delas Z, Plüss B, Ruiz-Dolz R. An Argumentation Scheme-Based Framework for Automatic Reconstruction of Natural Language Enthymemes. In: *Computational Models of Argument*; 2024. .
- [9] Ben-Naim J, David V, Hunter A. An Axiomatic Study of the Evaluation of Enthymeme Decoding in Weighted Structured Argumentation. In: *KR'25*; 2025. .
- [10] David V, Hunter A. A logic-based framework for decoding enthymemes in argument maps involving implicitness in premises and claims. In: *IJCAI*; 2025. .
- [11] Quan X, Valentino M, Dennis LA, Freitas A. Faithful and Robust LLM-Driven Theorem Proving for NLI Explanations. In: Che W, Nabende J, Shutova E, Pilehvar MT, editors. *Proceedings of the 63rd Annual Meeting of the ACL (Volume 1: Long Papers)*. Vienna, Austria: ACL; 2025. .
- [12] Nipkow T, Wenzel M, Paulson LC. *Isabelle/HOL: a proof assistant for higher-order logic*. Berlin, Heidelberg: Springer-Verlag; 2002.
- [13] Modgil S, Prakken H. The ASPIC+ framework for structured argumentation: a tutorial. *Argument & Computation*. 2014 Jan;5(1).
- [14] Prakken H, Wyner A, Bench-Capon T, Atkinson K. A formalization of argumentation schemes for legal case-based reasoning in ASPIC+. *Journal of Logic and Computation*. 2015 Oct;25(5).
- [15] Amgoud L, David V. Similarity Measures Based on Compiled Arguments. In: *ECSQARU'21*; 2021. .
- [16] Ruiz-Dolz R, Tavernier J, Lawrence J, Reed C. NLAS-multi: A multilingual corpus of automatically generated Natural Language Argumentation Schemes. *Data in Brief*. 2024;57.