Towards Fully Symbolic Knowledge Representation and Reasoning with Large Language Models: An Empirical Study of Formalisms for Neuro-Symbolic Question Answering

Marian-Sergiu Nistor *sergiunistor.info@gmail.com*
Independent Researcher

Abstract

Large language models store knowledge in their weights and reason through opaque associations, so their answers can be neither traced to supporting facts nor corrected. We show that neuro-symbolic question answering over a closed corpus is a practical alternative. In the proposed system, language models act only as translators: a corpus encoder turns the corpus into a Prolog knowledge base once, a query writer turns each question into a clause over it, and a symbolic engine derives the answer, so every answer is explained by a proof over facts that trace back to the corpus. We decompose answer F1 into *expressiveness*, the share of questions for which the query binds an answer, and *tractability*, the F1 of the answers it binds, and establish four results. First, the approach works for closed-domain question answering: across 84 representations, crossing 14 semantics, three groundings and two logics, with Claude Opus 5.5 and Haiku 5.5, the best representations clearly beat closed-book answering, grounding matters far more than semantics or logic, and on a category-diverse question set the pipeline outperforms retrieval-augmented generation (RAG) on held-out questions with Claude Opus, Sonnet and Haiku 5.5. Second, it improves factuality: it abstains when evidence is absent, stays faithful to counterfactual contexts, verifies claims more accurately than RAG and, with Opus, surpasses RAG on grounded long-form generation. Third, both translators can be trained: Qwen3-4B-Instruct, trained with supervised fine-tuning and reinforcement learning as corpus encoder and query writer, raises held-out F1 several-fold, transfers to unseen corpora and becomes more factual, although it is trained only for question answering. Fourth, the representation itself can be learned: a language model that rewrites the semantics from the system's failures improves F1 on the questions used for refinement in most runs and, in the best runs, on held-out questions and an unseen corpus. Confined to translation, a language model can thus drive a symbolic system in which every answer rests on a proof that can be checked.

1 Introduction

Large language models (LLMs) store factual knowledge implicitly in their parameters and derive answers from statistical associations acquired during pre-training, which remain opaque to inspection (Zhao et al., 2024; Bereska & Gavves, 2024). This implicit storage has three consequences. First, the knowledge that supports a given answer cannot be consistently located or audited: a fluent answer may rest on no supporting fact at all (Ji et al., 2023; Huang et al., 2025), and the sources a model cites frequently fail to support its claims (Liu et al., 2023; Gao et al., 2023). Second, a fact cannot be reliably updated, corrected or removed. Parametric knowledge is fixed at training time and becomes outdated as the world changes (Lazaridou et al., 2021; Dhingra et al., 2022); editing and unlearning methods intervene on the weights without guarantees, and knowledge believed to be removed can often be recovered from the model's internal representations or with limited fine-tuning (Patil et al., 2024; Łucki et al., 2024; Deeb & Roger, 2024; Gao et al., 2026; Lynch et al., 2024). Third, the reasoning that produces an answer is not observable: explanations generated alongside an answer, such as chains of thought, need not reflect the computation that actually produced it (Turpin et al., 2023; Lanham et al., 2023).

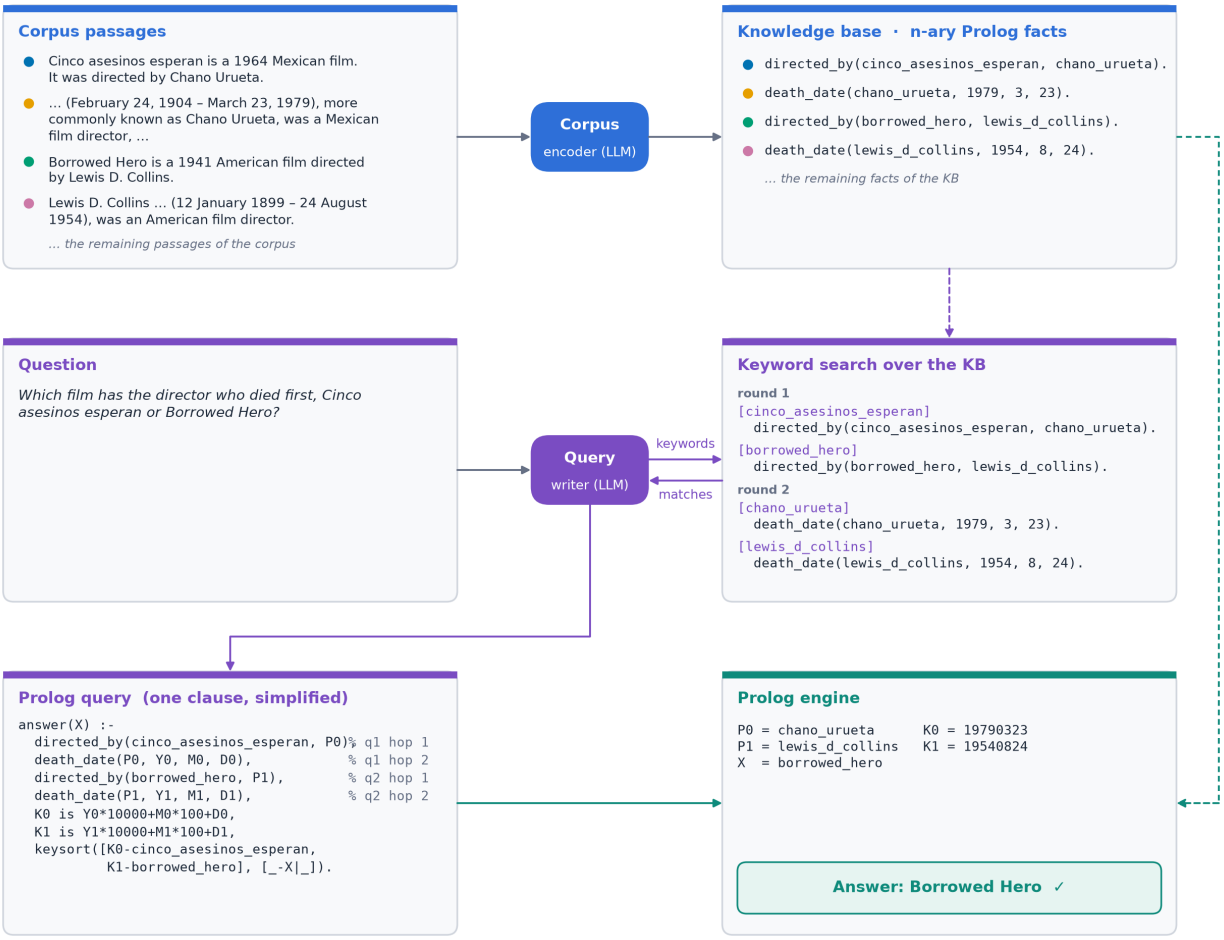


Figure 1: Overview of the pipeline on a multi-query question from 2WikiMultiHopQA (Ho et al., 2020), composed of two sub-queries of two hops each whose results are compared. A corpus encoder translates each passage into Prolog facts (here, under the n-ary semantics). Given the question, the query writer accesses the knowledge base only through keyword search and writes a single Prolog clause. Executing the clause over the knowledge base yields the answer, and the comparison between the two sub-queries is performed by the clause itself.

Neuro-symbolic methods offer a way to address these limitations, since they can separate the acquisition of knowledge from its use: neural networks can be exploited to extract symbolic concepts from data, while inference over those concepts is carried out by explicit symbolic procedures (Belle & Marcus, 2026; Yang et al., 2025b; Taddeo & Floridi, 2005; Mao et al., 2019). Rather than explaining an opaque model post hoc, such a system is interpretable by construction (Rudin, 2019). We adopt this view for question answering over a text corpus and restrict LLMs to the role of translators between natural language and a logic program. A corpus encoder translates each passage into Prolog facts and rules, yielding an executable knowledge base (KB). At inference, a query writer translates each question into Prolog queries. The answer is the set of bindings obtained by executing the queries, so every answer is derived by an inspectable proof over facts that can each be traced to the passage from which they were extracted. Moreover, because the knowledge resides in the KB rather than in the weights of either model, updating or removing a fact amounts to editing the KB, and the change applies to every subsequent question.

This transparency comes at a cost that has not been quantified, and it depends on the formalism into which the corpus is translated. Encoding a passage requires committing to a representation: a semantics,

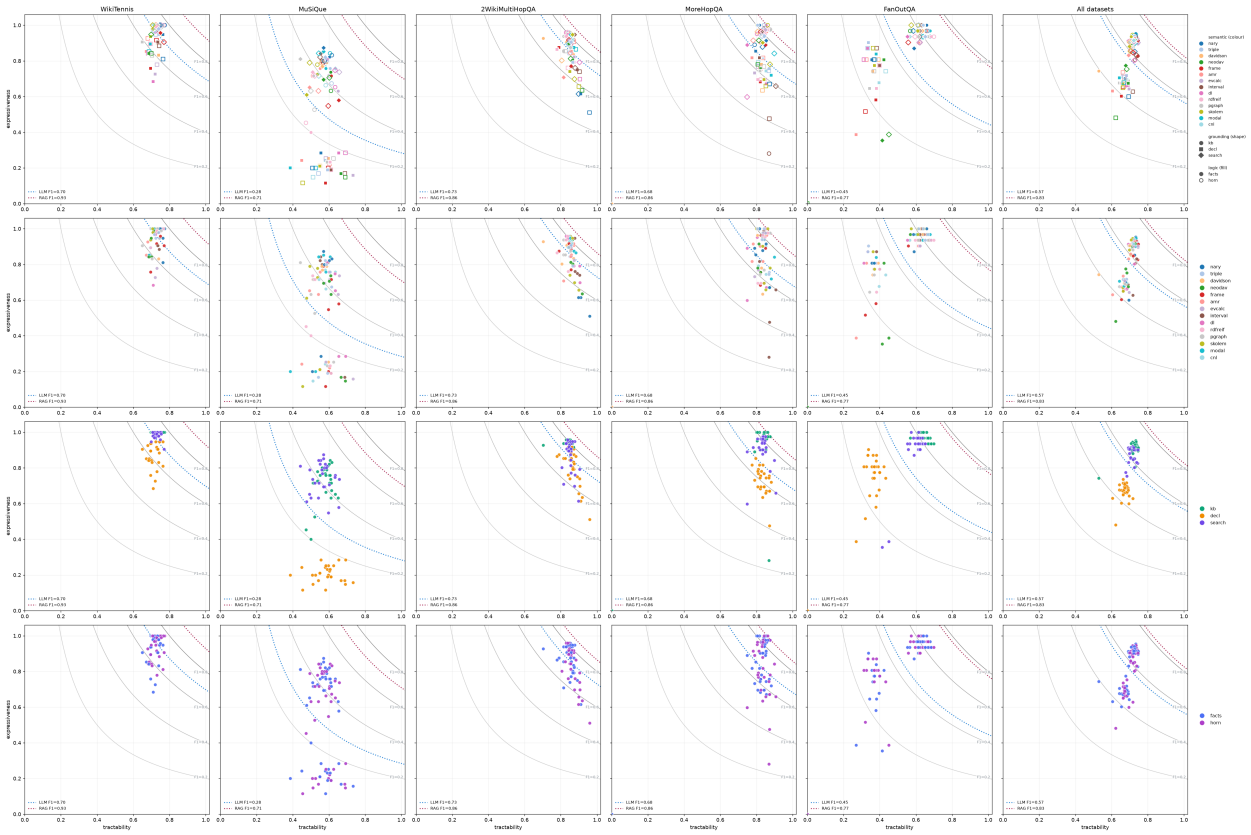


Figure 2: Expressiveness against tractability of every representation with Claude Opus 5.5 as corpus encoder and query writer, per dataset and pooled over all datasets. The top row distinguishes cells by color (semantic), marker shape (grounding), and fill (logic); the rows below color the same points by semantic, grounding, and logic. Dotted curves mark the F1 of the closed-book (LLM) and RAG baselines.

which determines how statements are decomposed into predicates and arguments; a grounding regime, which determines what the query writer sees of the KB: the KB itself, the declarations of its predicates, or only the results of searching it; and a logic, which admits either ground facts alone or also Horn rules. Each choice affects both whether a question can be expressed over the resulting KB and whether the answer is then retrieved correctly, an instance of the classical trade-off between expressiveness and tractability in knowledge representation (Levesque & Brachman, 1984).

We therefore report, alongside F1, two complementary measures: expressiveness, the proportion of questions for which the query binds an answer, and tractability, the F1 attained on those questions. Using Claude Opus 5.5 and Claude Haiku 5.5 as corpus encoders and query writers, we compare 84 representations, obtained by crossing 14 semantics, three grounding regimes and two logics, on five corpora with questions of one to four hops, answered by a single query or by several queries, and on three factuality benchmarks, against retrieval-augmented and closed-book baselines that use the same models.

We then ask whether the two translation tasks can be learned by an open-weight model rather than delegated to a large proprietary one. We train two separate instances of Qwen3-4B-Instruct (Yang et al., 2025a), each with its own LoRA adapter (Hu et al., 2021), through supervised fine-tuning followed by reinforcement learning: one serves as the corpus encoder and the other as the query writer for a given corpus. Training uses a subset of the corpus's questions, and the remainder is held out for evaluation. Every checkpoint is evaluated on the held-out questions of the training corpus and on the questions of the other corpora, which neither instance encounters during training. The two evaluations correspond to two use cases, both of which

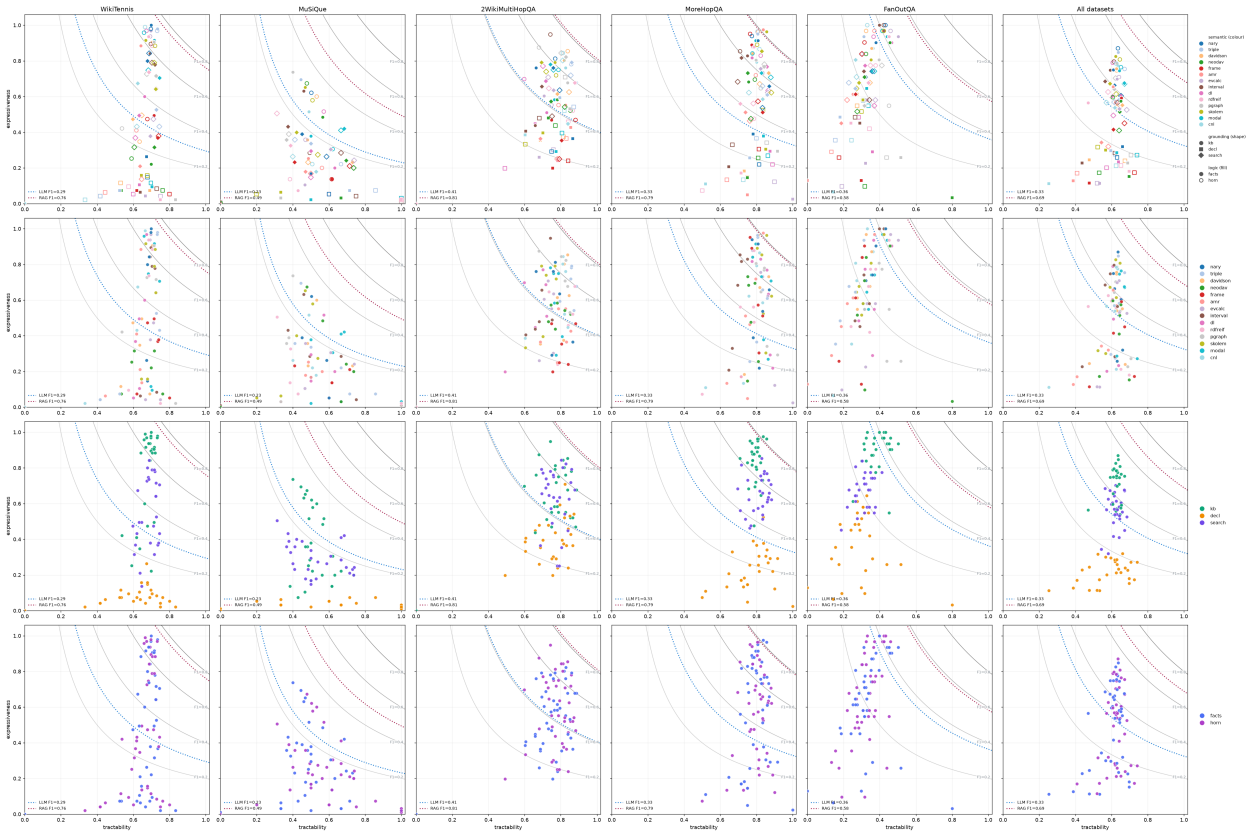


Figure 3: Expressiveness against tractability of every representation with Claude Haiku 5.5 as corpus encoder and query writer, per dataset and pooled over all datasets, with the same layout and baselines as Figure 2.

we consider valid. In the first case, the models are specialized to a single corpus: since that corpus is known in advance and encoded only once, fitting the models closely to it is acceptable, provided that they answer new questions about it. In the second case, the models serve as general translators that can be applied to corpora not seen during training. Every checkpoint is also evaluated on the three factuality benchmarks, none of which is used in training, to determine whether training for question answering changes how faithfully the models answer from the corpus.

Finally, we ask whether the representation itself can be learned. Starting from five of the best semantics and from an empty description, a language model rewrites the written semantics from the failures of the pipeline on a set of training questions, round after round, keeping the best version so far. We run this loop with Claude Opus, Sonnet and Haiku 5.5 as corpus encoder and query writer, and, with these fixed, vary the model that rewrites the semantics. As for the trained translators, we consider the specialized case, where the semantics is tuned to a known corpus and its questions, and evaluate every learned semantics on the targeted questions, on held-out questions and category combinations, and on a corpus never seen during learning.

Our contributions are as follows:

- We present a controlled comparison of 84 executable representations for question answering over a text corpus, obtained by crossing 14 semantics, three grounding regimes and two logics. We analyze how each of these choices affects expressiveness and tractability, across two model sizes, questions of one to four hops answered by a single query or by several queries, and corpora of increasing size.

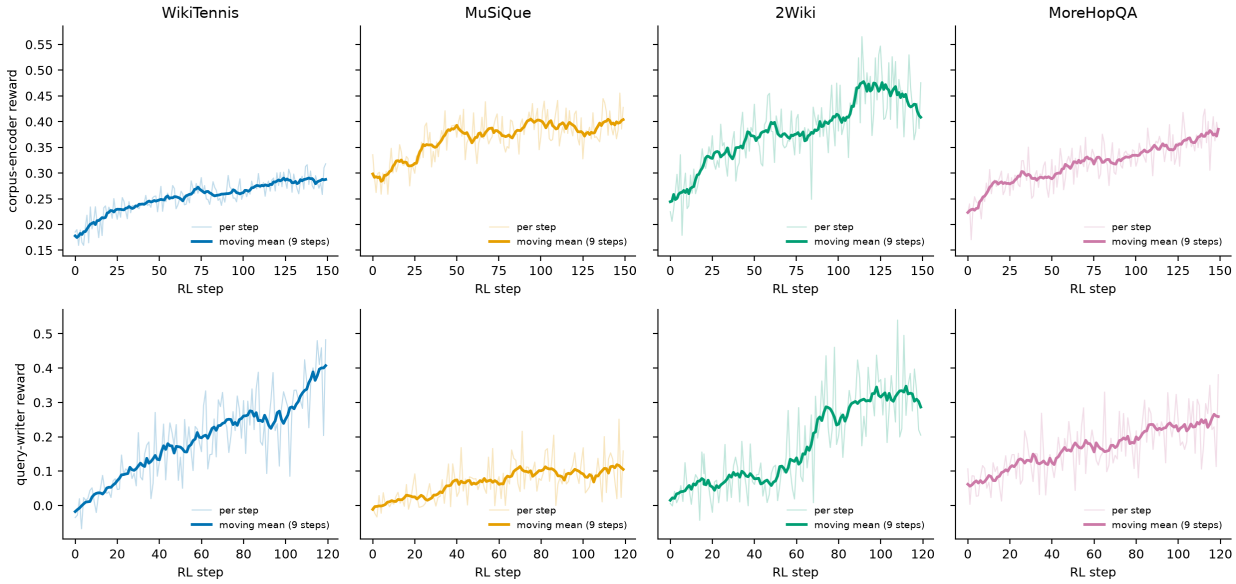


Figure 4: Reinforcement learning reward of the corpus encoder (top) and of the query writer (bottom) in each run, per step and as a moving mean over nine steps.

We also complement this quantitative analysis with a qualitative analysis of the knowledge bases and queries the models produce.

- We show that the resulting KBs answer from what the corpus states rather than from what the model believes. We validate this on three factuality benchmarks, against closed-book and retrieval-augmented baselines. On counterfactual contexts the pipeline remains faithful to the context, whereas closed-book models revert to their parametric knowledge.

- We show that both translation tasks can be learned by an open-weight model. Two instances of Qwen3-4B-Instruct, trained as corpus encoder and query writer through supervised fine-tuning followed by reinforcement learning, improve answer F1 several-fold on questions held out from training, and the improvement carries over to corpora not seen during training. Both expressiveness and tractability increase. We validate this with a split of each corpus's questions in which no held-out question shares a reasoning step with a training question, evaluating every training stage on the held-out questions and on the other corpora. Training also improves factuality, for which the models are never trained: on all three factuality benchmarks, the scores of the fully trained checkpoints rise above those of the untrained model.

- We show that the representation itself can be learned. A language model rewrites the written semantics from the failures of the pipeline on a set of training questions, keeping the best version so far, starting from five of the best semantics and from an empty specification. We run this loop with Claude Opus, Sonnet and Haiku 5.5 as corpus encoder and query writer, and, with the encoder and query writer fixed, vary the model that rewrites the semantics. On the targeted questions, the learned semantics improves F1 for every model, with most of the gain in the first two rounds, and a stronger rewriter finds improvements more reliably than a weaker one. We evaluate every learned semantics on held-out questions, held-out combinations of question categories and a corpus never seen during learning, where the best runs also improve.

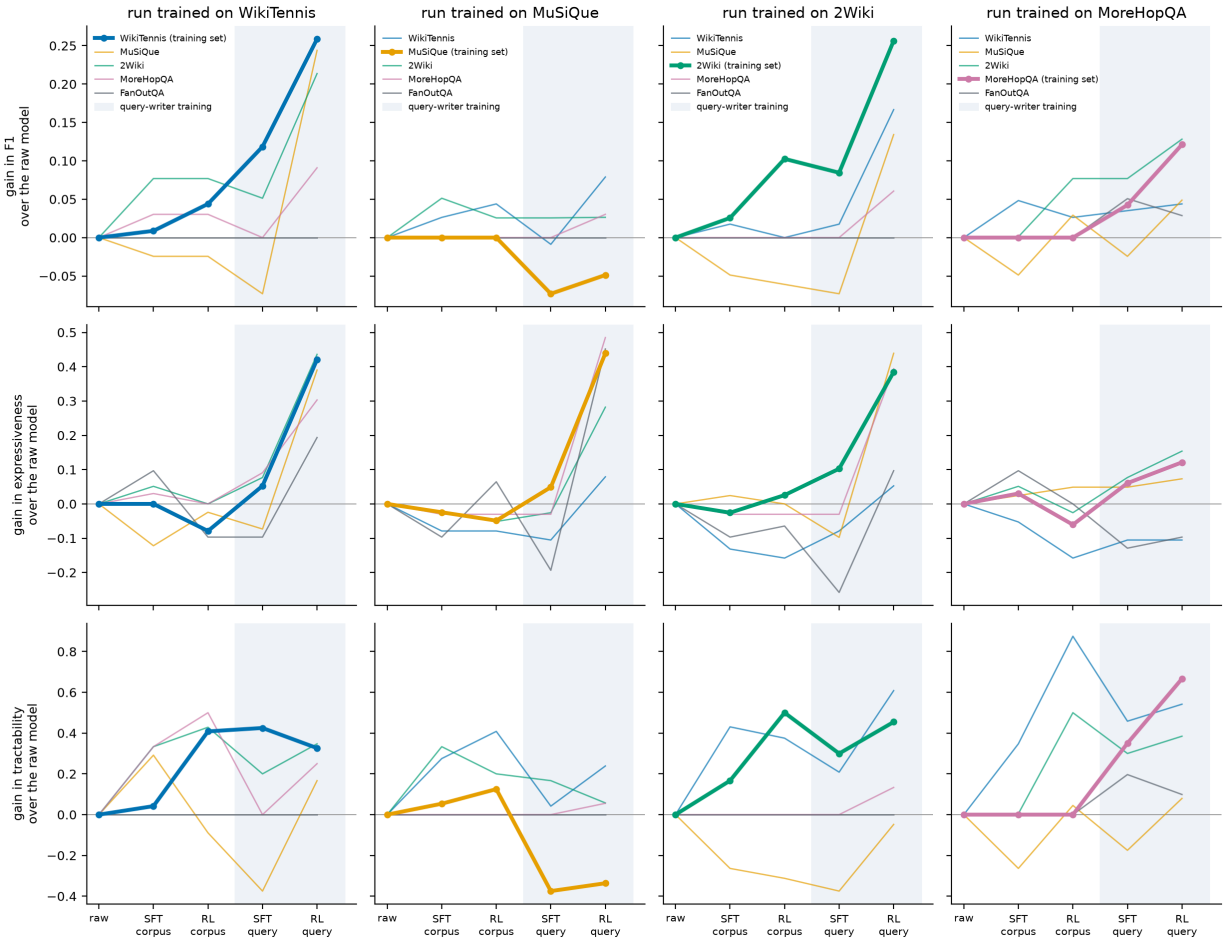


Figure 5: Gain over the base model in F1 (top), expressiveness (middle), and tractability (bottom) on the single-query test questions of each dataset after each training stage, per run. The thick line is the dataset the run was trained on, and the shaded stages train the query writer.

2 Related Work

LLMs as translators for symbolic solvers. A growing line of work uses LLMs to translate natural language into a formal language that an external solver then executes (Belle & Marcus, 2026). Logic-LM (Pan et al., 2023) translates a reasoning problem into first-order logic, and refines the translation using the error messages of the solver. LINC (Olausson et al., 2023) translates premises and conclusions into first-order logic and delegates inference to a theorem prover, and SatLM (Ye et al., 2023) has the model write a declarative specification of the problem that is solved by an SMT solver. These approaches formalize each problem together with the premises it provides: the formal representation is rebuilt for every input, and its premises are known in advance to be relevant to the question. Closest to our setting, π-CoT (Wan et al., 2025) addresses multi-hop question answering by translating the question into a Prolog query of single-hop sub-queries and, for each sub-query, having the LLM extract Prolog facts from the passages retrieved for it; the resulting Prolog answer is then supplied, together with the passages, to a final chain-of-thought prompt. Its KB is therefore built anew for every question, from facts extracted with that question in view. In contrast, we encode a corpus once into a KB that is shared by all questions and built without knowledge of them. The answer is the result of executing the query over the KB, with no subsequent generation by the LLM.



Figure 6: Gain over the base model in F1 (top), expressiveness (middle), and tractability (bottom) on the multi-query test questions of each dataset after each training stage, per run, with the same layout as Figure 5.

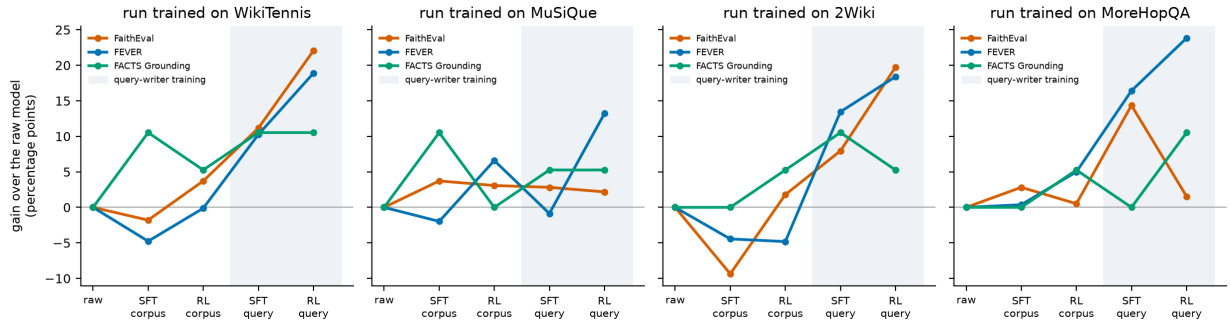


Figure 7: Gain over the base model, in percentage points, on FaithEval, FEVER, and FACTS Grounding after each training stage, per run. The shaded stages train the query writer.

Knowledge graph construction with LLMs. Another line of work uses LLMs to construct knowledge graphs from text, and differs mainly in how the schema of the graph is obtained. AutoSchemaKG (Bai et al.,

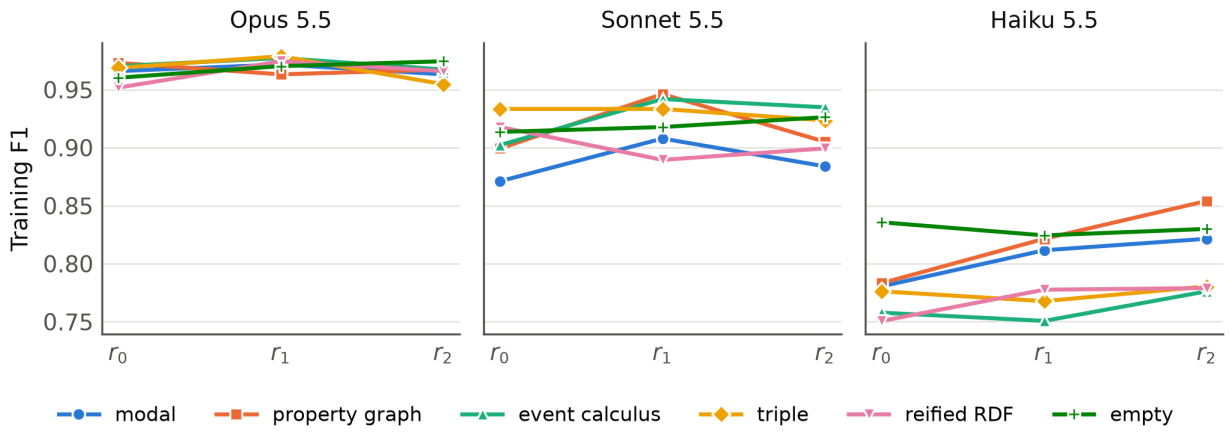


Figure 8: Training F1 after each round of rewriting, for each starting semantics, with Claude Opus, Sonnet and Haiku 5.5 as corpus encoder and query writer and Claude Opus 5.5 as the rewriter. Each round rewrites the best semantics found so far, so a round that scores lower is discarded.

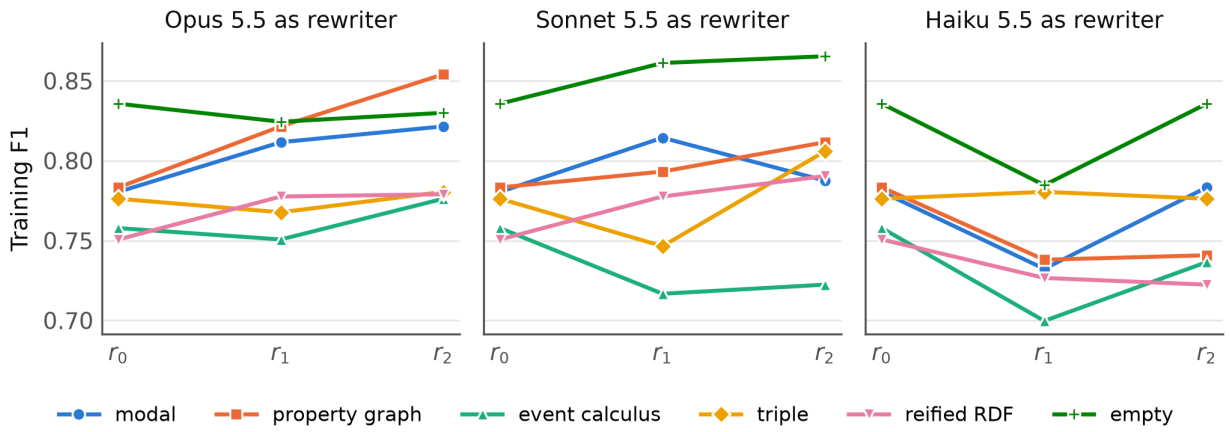


Figure 9: Training F1 after each round of rewriting, for each starting semantics, with Claude Haiku 5.5 as corpus encoder and query writer and Claude Opus, Sonnet and Haiku 5.5 as the rewriter.

2026) extracts triples between entities and events from web-scale corpora and induces their schema from the text, without a predefined ontology, and Chen et al. (2025) build hierarchical knowledge graphs from scientific articles through bottom-up extraction, splitting and abstraction. Other approaches instead ground the extraction in an ontology, either an existing one such as the Wikidata schema (Feng et al., 2024), or one first learned from the corpus and then used to guide the extraction prompts (Belfadel et al., 2026). These works evaluate the constructed graph intrinsically, through the quality of its triples or of its schema, or use it as a retrieval source for an LLM that generates the answer; AutoSchemaKG, for instance, is evaluated on multi-hop question answering by passing retrieved subgraphs to an LLM. In contrast, our KBs are executable logic programs, assessed by the questions they answer through execution alone.

Question answering over structured knowledge. Knowledge base question answering (KBQA) has long been framed as semantic parsing into an executable query, and recent systems assign this step to LLMs. ChatKBQA (Luo et al., 2024) fine-tunes an LLM to generate a logical form, grounds its entities

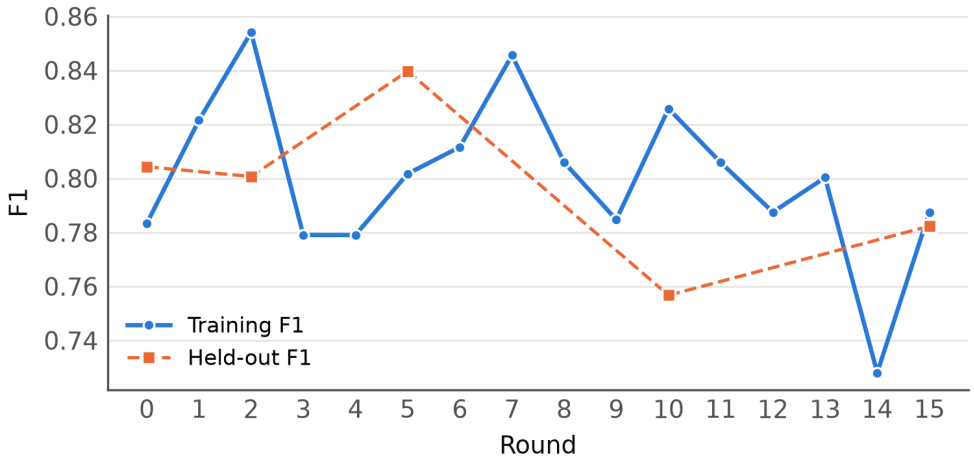


Figure 10: Training F1 over fifteen rounds of rewriting, and held-out F1 at rounds 0, 2, 5, 10 and 15, with Claude Haiku 5.5 as corpus encoder and query writer, the property graph as starting semantics and Claude Opus 5.5 as the rewriter.

and relations by retrieval, and executes it as a SPARQL query. Rule-KBQA (Zhang et al., 2025) learns a library of rules, partly generated by a rule-following fine-tuned LLM, and uses it to guide a symbolic agent that builds executable logical forms step by step. These systems query existing, curated KBs, so the vocabulary that a query must match is fixed in advance and independent of the model that writes it. A second family integrates structured knowledge without executing a query. OREO-LM (Hu et al., 2022) interleaves knowledge interaction layers with the layers of a language model, which perform a differentiable walk over a KB. Graph-based retrieval methods build a graph from the corpus with an LLM and use it to select evidence: GraphRAG (Edge et al., 2024) generates answers from summaries of communities of entities, and HippoRAG (Gutiérrez et al., 2024) ranks passages by personalized PageRank over an extracted graph before an LLM reads them. In both cases the answer is produced by a neural model and is not backed by a derivation over the retrieved knowledge. The query writer that we implement resembles a semantic parser, but its KB is built from the corpus by another model, so the query must match a vocabulary that the query writer did not choose.

Choice of representation. In knowledge representation, the choice of formalism has long been known to involve a trade-off: the more expressive the language, the less tractable inference is over it (Levesque & Brachman, 1984). Recent work revisits this question for LLMs. Beiser et al. (2025) show that the formal language into which an LLM translates a problem affects neuro-symbolic reasoning, comparing four languages across six LLMs on logical reasoning benchmarks. Wang et al. (2025) let the LLM select first-order logic, logic programming or Boolean satisfiability for each problem. Both studies vary the formal language into which the LLM translates, together with its solver, and translate each self-contained logic problem, premises and question together, in a single pass. Yu et al. (2026) compare fifteen serializations of a given knowledge graph and find that the format matters for structural reasoning but not for knowledge extraction, and Jin et al. (2024) provide LLMs with abstract meaning representations of their input as an intermediate step for reasoning. Other work argues for representations richer than triples, such as hyper-relational facts (Panda et al., 2024), propositions (Wang & Han, 2025) or triples augmented with context such as time, location and provenance (Xu et al., 2024). We instead compare representations for KBs that LLMs construct from a corpus, within a single pipeline in which the formal language is fixed. We vary three properties of the representation, namely the semantics, which determines how natural text is decomposed into predicates and arguments; the grounding, which determines what the query writer sees of the KB; and the logic, which admits either ground facts alone or also Horn rules, and measure the effect of each on both expressiveness and tractability.

Optimizing instructions from feedback. Our representation learning loop is related to methods that let a language model improve its own instructions from feedback. APE (Zhou et al., 2022) and OPRO (Yang et al., 2024) search over candidate instructions to maximize a score on example data, Pryzant et al. (2023) and TextGrad (Yuksekgonul et al., 2025) edit prompts and other components from natural language critiques of their errors, and DSPy (Khattab et al., 2024) optimizes the prompts and demonstrations of multi-stage language model pipelines toward a metric. Self-Refine (Madaan et al., 2023) has a model critique and revise its own output for a single input, and Reflexion (Shinn et al., 2023) has an agent keep verbal reflections in memory across repeated attempts at the same task; both improve one instance rather than a specification shared by all inputs. These methods optimize the instructions of a task whose output format is fixed. In our loop, the object being rewritten is the knowledge representation itself: the semantics that determines how every passage is encoded, and therefore what any later query can express.

3 Method

Figure 1 depicts the system architecture, and Algorithm 1 summarizes its procedure. It consists of two LLM roles and a symbolic engine. A corpus encoder reads the corpus a few passages at a time and translates each batch into logic program clauses in a target representation. The union of its outputs is a knowledge base, constructed before any question is posed. A query writer, which never sees the corpus, translates each question into a single Prolog clause over that knowledge base. The engine executes the clause, and the answers it binds are the answer of the system. No LLM sees the result of the execution or composes the final answer. Every answer is therefore the conclusion of a proof whose premises are knowledge base clauses, each produced from a known batch of passages. This design follows the neuro-symbolic methodology that uses LLMs as translators into a formal language executed by an external solver (Pan et al., 2023). When the semantics is learned rather than fixed, a third model, the rewriter, revises the written semantics between rounds (Section 3.5).

3.1 Representations

A representation is a triple $\mathcal{R} = (\sigma, \gamma, \lambda)$ of a semantics, a grounding and a logic. The semantics σ fixes the shape of a fact (e.g., `won(serena_williams, wimbledon)` with a predicate per relation, or `stmt(s1, serena_williams, won, wimbledon)` with the relation as an argument). The grounding γ fixes how the query writer comes to know the vocabulary of the knowledge base: by reading the knowledge base itself, by reading declarations of its predicates, or only by searching it. The logic λ fixes whether the knowledge base contains ground facts only, or ground facts together with Horn rules. The system is defined for any $\mathcal{R}$. The corpus encoder receives a written description of $\mathcal{R}$ and the query writer a description of its semantics; both receive the same conventions for naming constants, so the two roles agree on the target language.

3.2 Corpus Encoder

Let the corpus $D = \{d_1, \ldots, d_n\}$ be a set of passages, partitioned into batches $B_1, \ldots, B_m$ of at most b passages each. The corpus encoder E maps each batch to a set of clauses, and the knowledge base is their union:

$$K \;=\; \bigcup_{j=1}^{m} E(B_j; \mathcal{R}), \qquad E(B_j; \mathcal{R}) \;=\; F_j \cup H_j, \tag{1}$$

where F_j is a set of ground facts and H_j a set of Horn rules ($H_j = \emptyset$ when λ permits facts only). The corpus encoder is instructed to state only what its batch asserts, without adding world knowledge, and to encode exhaustively, since it does not know which questions will be asked. Identifiers that $\mathcal{R}$ introduces for events, statements or other nodes are derived from the passage identifier, so that they remain unique across batches. There is no separate entity resolution step: two batches refer to the same entity only when the corpus encoder spells its constant identically in both, which the naming conventions of $\mathcal{R}$ are designed to ensure. Clauses that do not parse are discarded before execution. Encoding batch by batch resembles the chunk-level extraction used to build graphs for retrieval and knowledge graphs from text (Edge et al., 2024; Bai et al., 2026); here, however, the output is an executable logic program.

3.3 Query Writer

The query writer Q has three inputs: the question q, as natural language text; a description of the semantics σ and of the naming conventions, so that it knows how facts are written; and the view $V_\gamma(K)$ of the knowledge base that the grounding γ permits, so that it knows what the knowledge base contains. Its output is a single Prolog clause c:

$$c \;=\; Q\big(q,\; \sigma,\; V_\gamma(K)\big). \tag{2}$$

The clause defines the predicate `answer(X)`: the values of X for which it holds are the answers to the question. Some questions require several lookups whose answers are combined, such as the union of the answers to several sub-questions, or a comparison among entities the question names. The combination is performed inside the single clause, rather than by a further model call. For a union, the clause takes the disjunction of the lookups. For a comparison, it binds a value from each lookup and returns the entity whose value is the earliest or the latest; if any value cannot be bound, the clause fails rather than defaulting to one of the candidates.

3.4 Query Execution

The engine computes the answer set of the clause over the knowledge base:

$$\hat{A}(q) \;=\; \big\{\, x \;\big|\; K \cup \{c\} \vdash \texttt{answer}(x) \,\big\}. \tag{3}$$

A clause that does not parse, raises an error or exceeds the time limit of the engine yields $\hat{A}(q) = \emptyset$. An empty answer set is an abstention: the system returns nothing rather than a guess. Two failures are therefore distinguished: a clause that binds nothing and a clause that binds a wrong answer.

3.5 Learning the Semantics

The corpus encoder receives the semantics only as a written description, so any such description can serve as a semantics, including one written by another model. We use this to learn the semantics from the failures of the pipeline rather than choose it from a fixed list. The loop starts either from one of the semantics of Table 1 or from an empty description that states only the conventions shared by every representation.

In each round, the corpus encoder translates every corpus with the current description, without seeing any question. The query writer then answers a set of training questions by searching the resulting knowledge base, and is given no description of the semantics, so the description can affect the answers only through the knowledge base built from it. The description is scored by the mean F1 of these answers.

A rewriter model then revises the best description found so far, following the prompt in Appendix D. It receives the scores of every description tried, the F1 for each question category, and the failed training questions with the search log of the query writer and the facts about their entities. Every revision is checked automatically before it is used: it may not contain the names of entities from the corpora, so that answers cannot be written into the semantics, and the rewriter must fix any revision that fails the check. A revision replaces the best description only if it scores higher, and the next round revises the best description again.

After the loop, the starting and the learned descriptions are each evaluated once on questions the loop never scored: held-out questions over the same corpora, questions that combine held-out categories, and a corpus that is never encoded during learning.

4 Experimental Setup

Our evaluation consists of three experiments. The first compares representations, varying the semantics, the grounding and the logic of the knowledge base with the same frontier LLMs in both roles, and measures how each choice affects question answering and factuality. The second asks whether two separately trained instances of an open-weight model can learn the roles of the corpus encoder and of the query writer, and whether what they learn transfers to corpora they were not trained on. The third asks whether the semantics

Algorithm 1 Question answering over an encoded corpus, with representation $\mathcal{R} = (\sigma, \gamma, \lambda)$.

function EncodeCorpus($D, \mathcal{R}, b$) ▷ offline, once per corpus
  $K \leftarrow \emptyset$
  for each batch $B \subseteq D$ with $|B| \leq b$ **do**
    $K \leftarrow K \cup \textsc{Parse}\big(E(B; \mathcal{R})\big)$ ▷ facts, and rules if λ permits
  end for
  return K
end function

function Answer($q, K, \mathcal{R}$) ▷ online, per question
  $c \leftarrow Q\big(q,\ \sigma,\ V_\gamma(K)\big)$ ▷ one clause defining `answer(X)`
  if c does not parse **then**
    return $\emptyset$ ▷ abstention
  end if
  return $\{x \mid K \cup \{c\} \vdash \texttt{answer}(x)\}$
end function

Table 1: The 14 semantics, each encoding "Serena Williams won Wimbledon in 2016".

semantics	example
n-ary	`won_title_at(serena_williams, wimbledon, 2016).`
triple	`won(w1, serena_williams). event_of(w1, wimbledon). year_of(w1, 2016).`
Davidsonian (Davidson, 1967)	`won(e1, serena_williams, wimbledon). time(e1, 2016).`
neo-Davidsonian	`event(e1, winning). agent(e1, serena_williams). theme(e1, wimbledon). time(e1, 2016).`
frame (Baker et al., 1998)	`frame(f1, competition_victory). fe(f1, winner, serena_williams). fe(f1, competition, wimbledon).`
AMR (Banarescu et al., 2013)	`instance(w1, win_01). edge(w1, arg0, serena_williams). edge(w1, arg1, wimbledon).`
event calculus (Kowalski & Sergot, 1986)	`happens(e1, win, 2016). role(e1, agent, serena_williams). role(e1, theme, wimbledon).`
interval (Allen, 1983)	`won(serena_williams, wimbledon, 2016, 2016).`
description logic (Baader et al., 2007)	`instance_of(serena_williams, tennis_player). won(serena_williams, w1). year_of(w1, 2016).`
reified RDF	`stmt(s1, serena_williams, won, wimbledon). meta(s1, year, 2016).`
property graph (Angles, 2018)	`node(serena_williams, player). edge(r1, serena_williams, wimbledon, won). prop(r1, year, 2016).`
Skolem (Skolem, 1920)	`beat_in_final(serena_williams, sk_opponent(v_2016_wimbledon)).`
modal	`won(serena_williams, wimbledon, 2016, stated).`
controlled NL (Kuhn, 2014)	`cnl(serena_williams, won_title_at, wimbledon).`

itself can be learned, by letting a model rewrite it from the failures of the pipeline. We execute each query clause with SWI-Prolog (Wielemaker et al., 2012), with a time limit of 2 seconds per query and at most 64 solutions, and score the solutions it returns.

4.1 Datasets

Question answering. All three experiments are evaluated on the same datasets, with the same measures and baselines. The five question answering corpora are MuSiQue (Trivedi et al., 2022), 2WikiMultiHopQA (Ho et al., 2020), MoreHopQA (Schnitzler et al., 2024), FanOutQA (Zhu et al., 2024), and WikiTennis, a

		Closed-book	RAG@3	`nary-kb-horn`			`nary-decl-facts`			`modal-search-facts`		
Dataset	Corpus	F1	F1	E	T	F1	E	T	F1	E	T	F1
Single-query												
WikiTennis	Original	0.698	0.928	1.000	0.774	0.774	0.947	0.763	0.723	1.000	0.714	0.714
	Tenfold		0.774	0.979	0.746	0.730	0.810	0.720	0.584	0.968	0.725	0.702
MuSiQue	Original	0.284	0.709	0.832	0.616	0.512	0.284	0.556	0.158	0.853	0.570	0.486
	Tenfold		0.530	0.716	0.515	0.368	0.116	0.455	0.053	0.716	0.583	0.417
2WikiMultiHopQA	Original	0.729	0.865	0.938	0.856	0.802	0.615	0.910	0.559	0.948	0.854	0.809
	Tenfold		0.865	0.615	0.848	0.521	0.583	0.851	0.496	0.938	0.856	0.802
MoreHopQA	Original	0.682	0.858	0.890	0.790	0.703	0.854	0.795	0.679	0.939	0.840	0.789
	Tenfold		0.854	0.976	0.838	0.817	0.598	0.837	0.500	0.927	0.851	0.789
FanOutQA	Original	0.446	0.774	0.935	0.681	0.637	0.806	0.357	0.288	0.968	0.604	0.584
	Tenfold		0.720	0.903	0.604	0.545	0.645	0.345	0.223	0.935	0.530	0.496
Mean	Original	0.568	0.827	0.919	0.743	0.686	0.701	0.676	0.481	0.941	0.716	0.676
	Tenfold		0.748	0.838	0.710	0.596	0.550	0.642	0.371	0.897	0.709	0.641
Multi-query												
WikiTennis	Original	0.744	0.852	1.000	0.816	0.816	0.994	0.820	0.815	0.988	0.827	0.816
	Tenfold		0.857	0.994	0.827	0.821	0.963	0.793	0.763	0.963	0.811	0.781
MuSiQue	Original	0.385	0.637	0.956	0.668	0.639	0.588	0.584	0.343	0.944	0.686	0.647
	Tenfold		0.625	0.906	0.583	0.528	0.400	0.496	0.198	0.950	0.601	0.571
2WikiMultiHopQA	Original	0.700	0.866	0.775	0.708	0.549	0.938	0.720	0.675	0.925	0.783	0.725
	Tenfold		0.820	0.825	0.815	0.672	0.887	0.696	0.618	0.988	0.712	0.703
MoreHopQA	Original	0.725	0.920	0.992	0.892	0.885	0.942	0.764	0.719	0.967	0.818	0.791
	Tenfold		0.924	0.825	0.770	0.635	0.892	0.723	0.644	0.967	0.801	0.775
Mean	Original	0.638	0.819	0.931	0.771	0.722	0.865	0.722	0.638	0.956	0.779	0.745
	Tenfold		0.806	0.887	0.749	0.664	0.785	0.677	0.556	0.967	0.732	0.708

Table 2: Expressiveness (E), tractability (T), and F1 of the baselines and of the best knowledge base, declared, and search cells with Claude Opus 5.5 on the original and tenfold corpora, averaged over the single-query and multi-query datasets.

corpus that we built for this study and describe below. Each benchmark presents a set of questions and their gold answers. We select the questions greedily so that as many as possible fit in a corpus of about 5,300 words (3,900 for MoreHopQA, whose pool is smaller), and a question is kept only if all of its supporting paragraphs are in the corpus. We also build a tenfold version of each corpus by adding 207 passages from the dataset's own pool to the 23 original ones (about 230 passages and 53,000 words). The questions and gold answers stay the same, so every question can still be answered from the original 23 passages, and the added ones only act as distractors. The tenfold corpora therefore test whether the system still finds the facts a question needs in a knowledge base that is ten times larger, mostly irrelevant to the question, and too large for the query writer to read in full.

A naturally connected corpus. In the four benchmark corpora, the passages are the supporting paragraphs of the selected questions, so each corpus is assembled around its questions; to test the system on a corpus that exists independently of the questions asked about it, we built WikiTennis, a multi-hop dataset over English Wikipedia. We first collected 419 linked Wikipedia articles about athletes, by starting from one article and repeatedly adding related ones, so that the articles share entities and their relations recur, and took from this collection the opening sections of 23 articles about tennis players. To write the questions, we divided these passages into eight groups of at most three and gave each group to Claude Opus 5.5 with the prompt in Appendix E, which asks for twelve questions of two to four hops each, with their answers and the passages that support them. The prompt requires every step of a question to be stated literally in the passages, and it asks for answers of varied types (entities, counts, years, places and categories) and for questions that combine several passages where the passages allow it.

		Closed-book	RAG@3	`nary-kb-horn`			`nary-decl-facts`			`modal-search-facts`		
Dataset	Corpus	F1	F1	E	T	F1	E	T	F1	E	T	F1
Single-query												
WikiTennis	Original	0.295	0.761	0.979	0.697	0.682	0.263	0.673	0.177	0.705	0.744	0.525
	Tenfold		0.753	0.958	0.709	0.679	0.147	0.798	0.117	0.505	0.694	0.351
MuSiQue	Original	0.232	0.494	0.621	0.503	0.312	0.032	0.667	0.021	0.421	0.683	0.288
	Tenfold		0.459	0.232	0.455	0.105	0.011	0.000	0.000	0.190	0.778	0.147
2WikiMultiHopQA	Original	0.406	0.812	0.812	0.769	0.625	0.365	0.867	0.316	0.781	0.742	0.580
	Tenfold		0.781	0.573	0.722	0.414	0.208	0.867	0.181	0.448	0.791	0.354
MoreHopQA	Original	0.329	0.793	0.939	0.775	0.728	0.390	0.781	0.305	0.732	0.828	0.606
	Tenfold		0.756	0.500	0.813	0.406	0.049	0.750	0.037	0.720	0.859	0.618
FanOutQA	Original	0.364	0.583	1.000	0.432	0.432	0.548	0.317	0.174	0.742	0.353	0.262
	Tenfold		0.565	0.613	0.354	0.217	0.516	0.276	0.142	0.645	0.342	0.221
Mean	Original	0.325	0.689	0.870	0.635	0.556	0.320	0.661	0.199	0.676	0.670	0.452
	Tenfold		0.663	0.575	0.610	0.364	0.186	0.538	0.095	0.501	0.693	0.338
Multi-query												
WikiTennis	Original	0.285	0.804	1.000	0.779	0.779	0.231	0.442	0.102	0.875	0.682	0.597
	Tenfold		0.780	0.938	0.760	0.712	0.269	0.545	0.146	0.775	0.647	0.502
MuSiQue	Original	0.277	0.529	0.294	0.484	0.142	0.094	0.429	0.040	0.700	0.424	0.297
	Tenfold		0.511	0.600	0.520	0.312	0.175	0.403	0.070	0.406	0.576	0.234
2WikiMultiHopQA	Original	0.246	0.791	0.887	0.766	0.680	0.575	0.639	0.367	0.800	0.690	0.552
	Tenfold		0.715	0.825	0.593	0.489	0.450	0.704	0.317	0.762	0.660	0.503
MoreHopQA	Original	0.246	0.829	0.892	0.712	0.634	0.817	0.613	0.500	0.892	0.741	0.661
	Tenfold		0.895	0.433	0.779	0.338	0.325	0.714	0.232	0.908	0.711	0.645
Mean	Original	0.263	0.738	0.768	0.685	0.559	0.429	0.531	0.253	0.817	0.634	0.527
	Tenfold		0.725	0.699	0.663	0.463	0.305	0.591	0.191	0.713	0.649	0.471

Table 3: Expressiveness (E), tractability (T), and F1 of the baselines and of the best knowledge base, declared, and search cells with Claude Haiku 5.5 on the original and tenfold corpora, averaged over the single-query and multi-query datasets.

Multi-query questions. Except for those of FanOutQA, each of the questions above is answered by a single multi-hop query; to evaluate questions whose answer requires the results of several independent queries, we additionally construct a multi-query question set over the same corpora. Each multi-query question combines one to four sub-questions, and each sub-question takes one to four hops to answer. The sub-questions come from the corpus's own questions: a sub-question is either one of these questions or a shorter version of it that stops after its first hops, whose answer the dataset already records. Most multi-query questions are unions, whose answer is the set of answers to all of their sub-questions; on 2WikiMultiHopQA, whose questions often end in dates, some are instead comparisons, which name several entities and ask which of them has the earliest or the latest date. A script selects the combinations of sub-questions and computes every gold answer from the dataset deterministically. We then use Claude Opus 5.5 only to write the question wording, with the prompt in Appendix F. For WikiTennis and MuSiQue, we draw ten questions for every combination of the number of sub-questions and the number of hops, which gives 160 questions each; 2WikiMultiHopQA, which has few questions with more than two hops, gets 80 questions of one or two hops, and MoreHopQA gets 120 questions of up to three hops. FanOutQA needs no such construction, since each of its questions already asks for one value for each of four to six entities.

Factuality. We evaluate factuality on three benchmarks: FaithEval (Ming et al., 2025), which tests faithfulness to a context that may be counterfactual, inconsistent or insufficient to answer the question; FEVER (Thorne et al., 2018), which tests the verification of claims against evidence; and FACTS Grounding (Jacovi et al., 2025), which tests long-form answers grounded in a document. As for question answering, the items of each benchmark are subsampled so that their contexts fit in a corpus of about 5,300 words. On FACTS Grounding, the facts bound by the clause are passed to a separate instance of the model, which writes the

	FaithEval			FEVER			FACTS
Arm	E	T	F1	E	T	F1	Score
Closed-book	0.885	0.174	0.231	0.933	0.706	0.726	0.000
RAG@3	0.654	0.941	0.962	0.696	0.904	0.896	0.737
`nary-kb-horn`	0.462	0.847	0.737	0.578	0.936	0.837	0.789
`triple-kb-facts`	0.577	0.744	0.776	0.548	0.959	0.837	0.895
`skolem-kb-horn`	0.500	0.897	0.795	0.593	0.963	0.881	0.842
`skolem-kb-facts`	0.462	0.972	0.795	0.578	0.962	0.867	0.737
`triple-kb-horn`	0.500	0.859	0.776	0.541	0.959	0.830	0.632
`nary-decl-facts`	0.385	0.850	0.673	0.422	0.947	0.711	0.632
`pgraph-decl-horn`	0.346	0.963	0.679	0.267	0.889	0.541	0.421
`triple-decl-horn`	0.385	0.817	0.660	0.511	0.913	0.756	0.316
`triple-decl-facts`	0.385	0.817	0.660	0.474	0.922	0.733	0.316
`modal-decl-facts`	0.308	0.958	0.641	0.489	0.939	0.763	0.421
`modal-search-facts`	0.500	0.897	0.795	0.556	0.947	0.837	0.842
`modal-search-horn`	0.423	0.970	0.756	0.585	0.949	0.859	0.789
`pgraph-search-horn`	0.346	0.852	0.641	0.607	0.927	0.852	0.474
`nary-search-facts`	0.423	0.924	0.737	0.593	0.938	0.852	0.789
`interval-search-facts`	0.385	0.800	0.654	0.570	0.961	0.867	0.737

Table 4: Expressiveness (E), tractability (T), and F1 on FaithEval and FEVER, and the FACTS Grounding score, of the baselines and the five best cells of each grounding with Claude Opus 5.5.

	FaithEval			FEVER			FACTS
Arm	E	T	F1	E	T	F1	Score
Closed-book	0.769	0.100	0.192	0.919	0.637	0.607	0.000
RAG@3	0.692	0.778	0.846	0.704	0.905	0.904	0.737
`nary-kb-horn`	0.500	0.808	0.750	0.526	0.930	0.800	0.263
`triple-kb-facts`	0.385	0.883	0.686	0.467	0.921	0.733	0.421
`skolem-kb-horn`	0.462	0.958	0.788	0.519	0.929	0.785	0.421
`skolem-kb-facts`	0.423	0.955	0.750	0.519	0.943	0.807	0.526
`triple-kb-horn`	0.423	0.803	0.686	0.519	0.929	0.785	0.684
`nary-decl-facts`	0.154	0.750	0.462	0.267	0.944	0.578	0.158
`pgraph-decl-horn`	0.038	1.000	0.385	0.207	0.857	0.504	0.105
`triple-decl-horn`	0.154	0.833	0.474	0.200	0.926	0.511	0.263
`triple-decl-facts`	0.231	0.889	0.551	0.185	0.960	0.504	0.158
`modal-decl-facts`	0.231	1.000	0.577	0.237	0.969	0.556	0.053
`modal-search-facts`	0.423	0.727	0.615	0.489	0.909	0.741	0.526
`modal-search-horn`	0.385	0.750	0.635	0.489	0.924	0.756	0.368
`pgraph-search-horn`	0.423	0.545	0.577	0.504	0.912	0.748	0.211
`nary-search-facts`	0.500	0.577	0.596	0.556	0.933	0.830	0.368
`interval-search-facts`	0.385	0.817	0.660	0.452	0.951	0.741	0.632

Table 5: Expressiveness (E), tractability (T), and F1 on FaithEval and FEVER, and the FACTS Grounding score, of the baselines and the five best cells of each grounding with Claude Haiku 5.5.

long-form answer from those facts alone, without the document, using the prompt in Appendix G. Following the benchmark, using the prompt proposed in the original paper, a judge marks each answer as eligible (it addresses the request) and grounded (every claim is supported by the document), and the score is the share of answers that are both.

	FaithEval				FEVER			
	Answered		Abstained		Answered		Abstained	
Arm	Right	Wrong	Right	Wrong	Right	Wrong	Right	Wrong
Closed-book	0.154	0.731	0.077	0.038	0.659	0.274	0.067	0.000
RAG@3	0.615	0.038	0.346	0.000	0.630	0.067	0.267	0.037
`nary-kb-horn`	0.391	0.071	0.346	0.192	0.541	0.037	0.296	0.126
`triple-kb-facts`	0.429	0.147	0.346	0.077	0.526	0.022	0.311	0.141
`skolem-kb-horn`	0.449	0.051	0.346	0.154	0.570	0.022	0.311	0.096
`skolem-kb-facts`	0.449	0.013	0.346	0.192	0.556	0.022	0.311	0.111
`triple-kb-horn`	0.429	0.071	0.346	0.154	0.519	0.022	0.311	0.148
`nary-decl-facts`	0.327	0.058	0.346	0.269	0.400	0.022	0.311	0.267
`pgraph-decl-horn`	0.333	0.013	0.346	0.308	0.237	0.030	0.304	0.430
`triple-decl-horn`	0.314	0.071	0.346	0.269	0.467	0.044	0.289	0.200
`triple-decl-facts`	0.314	0.071	0.346	0.269	0.437	0.037	0.296	0.230
`modal-decl-facts`	0.295	0.013	0.346	0.346	0.459	0.030	0.304	0.207
`modal-search-facts`	0.449	0.051	0.346	0.154	0.526	0.030	0.311	0.133
`modal-search-horn`	0.410	0.013	0.346	0.231	0.556	0.030	0.304	0.111
`pgraph-search-horn`	0.295	0.051	0.346	0.308	0.563	0.044	0.289	0.104
`nary-search-facts`	0.391	0.032	0.346	0.231	0.556	0.037	0.296	0.111
`interval-search-facts`	0.308	0.077	0.346	0.269	0.548	0.022	0.319	0.111

Table 6: Share of FaithEval and FEVER items that each arm answered correctly or incorrectly and on which it abstained correctly (gold label unknown) or incorrectly, with Claude Opus 5.5. The four shares sum to one, and F1 equals the sum of the two correct shares.

A category-diverse question set. The benchmark questions are dominated by bridge chains and comparisons, so for learning the semantics we built a question set that covers the kinds of reasoning a knowledge base should support. We defined 23 question categories in eight groups: lookup and structure, comparison and aggregation, time, logic, uncertainty and attribution, causation, procedures, and answerability (Appendix B). Each category is crossed with one to four hops and with single or multiple sub-queries. Claude Opus 5.5 generated 1,991 questions over the passages, a separate instance answered each question without seeing its gold answer, and disagreements were adjudicated, which kept 1,987 questions. We also tagged the benchmark questions with these categories and removed 126 of 919 whose gold answer was wrong or not supported by the passages. Because the benchmark corpora rarely contain procedures, defaults or conflicting sources, we added two corpora of real text, on British royal succession and on procedures and recipes, so that every category has at least 30 questions. Six corpora are used for learning: WikiTennis, MuSiQue, 2WikiMultiHopQA, FanOutQA and the two new corpora. MoreHopQA is never used for learning. The questions are split into 353 training questions, with at least 30 per category, and three held-out sets: 185 new questions over the six corpora, with at least 20 per category; 66 questions that combine two categories never seen together in training (superlatives with event order, and inverse relations with succession); and the 454 questions of MoreHopQA.

4.2 Measures

Each question is scored by the F1 between the set of answers the clause binds and the set of gold answers. In question answering, where every question has an answer, an unbound clause scores zero, so the mean F1 over a question set factors into two measures, which we name following the terminology defined in Levesque & Brachman (1984):

$$\mathrm{F1} \;=\; \underbrace{\frac{|\{q : \hat{A}(q) \neq \emptyset\}|}{|\mathcal{Q}|}}_{\text{expressiveness}} \;\times\; \underbrace{\frac{1}{|\{q : \hat{A}(q) \neq \emptyset\}|} \sum_{q:\hat{A}(q)\neq\emptyset} \mathrm{F1}(q)}_{\text{tractability}} . \tag{4}$$

	FaithEval				FEVER			
	Answered		Abstained		Answered		Abstained	
Arm	Right	Wrong	Right	Wrong	Right	Wrong	Right	Wrong
Closed-book	0.077	0.692	0.115	0.115	0.585	0.333	0.022	0.059
RAG@3	0.538	0.154	0.308	0.000	0.637	0.067	0.267	0.030
`nary-kb-horn`	0.404	0.096	0.346	0.154	0.489	0.037	0.311	0.163
`triple-kb-facts`	0.340	0.045	0.346	0.269	0.430	0.037	0.304	0.230
`skolem-kb-horn`	0.442	0.019	0.346	0.192	0.481	0.037	0.304	0.178
`skolem-kb-facts`	0.404	0.019	0.346	0.231	0.489	0.030	0.319	0.163
`triple-kb-horn`	0.340	0.083	0.346	0.231	0.481	0.037	0.304	0.178
`nary-decl-facts`	0.115	0.038	0.346	0.500	0.252	0.015	0.326	0.407
`pgraph-decl-horn`	0.038	0.000	0.346	0.615	0.178	0.030	0.326	0.467
`triple-decl-horn`	0.128	0.026	0.346	0.500	0.185	0.015	0.326	0.474
`triple-decl-facts`	0.205	0.026	0.346	0.423	0.178	0.007	0.326	0.489
`modal-decl-facts`	0.231	0.000	0.346	0.423	0.230	0.007	0.326	0.437
`modal-search-facts`	0.308	0.115	0.308	0.269	0.444	0.044	0.296	0.215
`modal-search-horn`	0.288	0.096	0.346	0.269	0.452	0.037	0.304	0.207
`pgraph-search-horn`	0.231	0.192	0.346	0.231	0.459	0.044	0.289	0.207
`nary-search-facts`	0.288	0.212	0.308	0.192	0.519	0.037	0.311	0.133
`interval-search-facts`	0.314	0.071	0.346	0.269	0.430	0.022	0.311	0.237

Table 7: Share of FaithEval and FEVER items that each arm answered correctly or incorrectly and on which it abstained correctly (gold label unknown) or incorrectly, with Claude Haiku 5.5. The four shares sum to one, and F1 equals the sum of the two correct shares.

		Training	Held-out F1			Held-out overall		
Model	Arm	F1	Same corpora	Category pairs	Unseen corpus	E	T	F1
Opus	Closed-book	0.786	0.749	0.712	0.821	0.935	0.884	0.791
	RAG@3	0.888	0.951	0.841	0.827	0.907	0.940	0.861
	Pipeline, start	0.965	0.950	0.955	0.977	0.990	0.978	0.968
Sonnet	Closed-book	0.734	0.700	0.720	0.717	0.893	0.839	0.713
	RAG@3	0.926	0.941	0.886	0.868	0.932	0.948	0.889
	Pipeline, start	0.906	0.891	0.895	0.944	0.982	0.949	0.926
Haiku	Closed-book	0.465	0.551	0.242	0.360	0.468	0.822	0.399
	RAG@3	0.895	0.900	0.606	0.804	0.865	0.926	0.811
	Pipeline, start	0.781	0.786	0.775	0.848	0.915	0.902	0.825

Table 8: F1 of the closed-book and RAG@3 baselines and of the pipeline on the category-diverse question set, for each model as corpus encoder and query writer. The pipeline row averages the six starting semantics before any rewriting. The three held-out sets are new questions over the six training corpora (Same corpora, 185 questions), questions combining two categories never seen together in training (Category pairs, 66), and MoreHopQA, which is never used for learning (Unseen corpus, 454). Expressiveness (E), tractability (T) and F1 are pooled over all 705 held-out questions.

Expressiveness is the share of questions for which the clause binds any answer: it requires that the representation can state what the question asks, that the corpus encoder put the fact into the knowledge base, and that the query matches the names the corpus encoder chose. Tractability is the mean F1 over the questions that are answered: given that the system answers, how often it answers correctly. An inappropriate representation can be expressive but intractable, answering often and wrongly, or tractable but inexpressive, answering rarely and correctly; the two call for different remedies.

Model	Start	Training F1			Best	Held-out F1		
		r_0	r_1	r_2		Start	Best	Δ
Opus	modal	0.966	**0.972**	0.963	r_1	0.964	0.966	+0.002
	property graph	**0.973**	0.963	0.967	r_0	0.971	–	–
	event calculus	0.970	**0.977**	0.967	r_1	0.967	0.975	+0.008
	triple	0.969	**0.979**	0.955	r_1	0.969	0.970	+0.001
	reified RDF	0.952	**0.975**	0.966	r_1	0.970	0.976	+0.006
	empty	0.960	0.970	**0.975**	r_2	0.966	0.971	+0.005
Sonnet	modal	0.871	**0.908**	0.884	r_1	0.921	0.912	−0.009
	property graph	0.899	**0.946**	0.905	r_1	0.945	0.943	−0.001
	event calculus	0.902	**0.942**	0.935	r_1	0.925	0.939	+0.014
	triple	**0.933**	0.933	0.923	r_0	0.925	–	–
	reified RDF	**0.918**	0.889	0.899	r_0	0.929	–	–
	empty	0.914	0.918	**0.926**	r_2	0.911	0.939	+0.028
Haiku	modal	0.780	0.812	**0.822**	r_2	0.838	0.849	+0.011
	property graph	0.783	0.822	**0.854**	r_2	0.804	0.801	−0.004
	event calculus	0.758	0.751	**0.776**	r_2	0.806	0.834	+0.028
	triple	0.776	0.768	**0.780**	r_2	0.828	0.721	−0.107
	reified RDF	0.751	0.778	**0.779**	r_2	0.782	0.773	−0.009
	empty	**0.836**	0.824	0.830	r_0	0.891	–	–

Table 9: Training F1 of each run after each round of rewriting, with the best round in bold, and held-out F1 of the starting and of the best semantics, for each model as corpus encoder and query writer. Claude Opus 5.5 rewrites the semantics. When the starting semantics stays best, the held-out sets are not evaluated again.

Setting	Model	Training gain	Held-out gain			
			Same corpora	Category pairs	Unseen corpus	Overall
Encoder and query writer	Opus	+0.010	+0.008	−0.011	+0.004	+0.004
	Sonnet	+0.023	+0.006	+0.028	+0.002	+0.005
	Haiku	+0.027	−0.021	−0.096	+0.002	−0.013
Rewriter	Opus	+0.027	−0.021	−0.096	+0.002	−0.013
	Sonnet	+0.027	−0.022	−0.058	−0.035	−0.034
	Haiku	+0.001	−0.010	+0.004	−0.002	−0.004

Table 10: Mean gain of the best semantics over the starting semantics across the six starting points, on the training questions and on each held-out set. A run whose starting semantics stays best counts as zero gain. In the upper block, the model is the corpus encoder and query writer and Claude Opus 5.5 rewrites; in the lower block, Claude Haiku 5.5 is the corpus encoder and query writer and the model rewrites.

On FaithEval and FEVER, the gold answer may be unknown, and a clause that binds nothing answers unknown, which is correct when the gold label is unknown, so the mean F1 no longer factors as in Equation 4. A solution counts as an answer only if it is a value, rather than a placeholder that the corpus encoder created for an unnamed entity, an unbound variable or a whole fact term. Each answer is then checked by a separate instance of the same model, which sees the question, the clause, its answers and the facts and rules of the knowledge base that their derivation used, but neither the passage nor the gold answer; an answer it does not find established by those facts counts as an abstention. For Qwen, which we only run in training, the check is made by Claude Haiku 5.5. We therefore also divide the items into correct and incorrect answers and correct and incorrect abstentions, whose two correct shares sum to the F1.

In the category-diverse question set, some gold answers are free text, such as a reason or the steps of a procedure. These answers, and any answer whose set F1 is below one, are graded by Claude Opus 5.5 as

Rewriter	Start	Training F1			Best	Held-out F1		
		r_0	r_1	r_2		Start	Best	Δ
Opus	modal	0.780	0.812	**0.822**	r_2	0.838	0.849	+0.011
	property graph	0.783	0.822	**0.854**	r_2	0.804	0.801	−0.004
	event calculus	0.758	0.751	**0.776**	r_2	0.806	0.834	+0.028
	triple	0.776	0.768	**0.780**	r_2	0.828	0.721	−0.107
	reified RDF	0.751	0.778	**0.779**	r_2	0.782	0.773	−0.009
	empty	**0.836**	0.824	0.830	r_0	0.891	–	–
Sonnet	modal	0.780	**0.814**	0.787	r_1	0.838	0.821	−0.016
	property graph	0.783	0.793	**0.812**	r_2	0.804	0.647	−0.157
	event calculus	**0.758**	0.717	0.722	r_0	0.806	–	–
	triple	0.776	0.747	**0.806**	r_2	0.828	0.823	−0.005
	reified RDF	0.751	0.778	**0.790**	r_2	0.782	0.816	+0.034
	empty	0.836	0.861	**0.865**	r_2	0.891	0.833	−0.058
Haiku	modal	0.780	0.732	**0.783**	r_2	0.838	0.831	−0.006
	property graph	**0.783**	0.738	0.741	r_0	0.804	–	–
	event calculus	**0.758**	0.700	0.737	r_0	0.806	–	–
	triple	0.776	**0.780**	0.776	r_1	0.828	0.812	−0.016
	reified RDF	**0.751**	0.727	0.722	r_0	0.782	–	–
	empty	**0.836**	0.785	0.836	r_0	0.891	–	–

Table 11: Training and held-out F1 of each run with Claude Haiku 5.5 as corpus encoder and query writer, for each model as the rewriter. The rows with Opus as rewriter repeat the Haiku rows of Table 9.

a judge, which assigns 1 when the answers convey the gold answer, 0.5 when they convey part of it, and 0 otherwise, following fixed rules for competing values, multi-part questions and missing precision. The judge sees neither the passages nor the knowledge base. Over the 2,709 answers that were judged more than once across runs, it assigned the same score in 98.9% of the pairs.

4.3 Baselines

We compare against two baselines that use the same model as the pipeline. In the closed-book baseline, the model answers from the question alone, which measures what it already knows. In RAG@3, it answers from the question and the three passages ranked highest by BM25 (Robertson & Zaragoza, 2009), as many as the corpus encoder sees at a time, and both baselines are scored in the same way as the pipeline. On the category-diverse question set, both baselines are run with each of the three models on the training and held-out questions, retrieving from the corpus that the question is about.

4.4 Comparing Knowledge Representations

Semantics. We compare the 14 semantics listed in Table 1, each of which represents the same statement in a different way.

Groundings. The grounding determines what the query writer is shown of the knowledge base. With `kb`, it sees the whole knowledge base, and can copy exact predicate names and constants. With `decl`, it sees only the declarations the corpus encoder wrote for its predicates, and must guess the spelling of constants. With `search`, it sees nothing up front and can only query a keyword search over the knowledge base, which returns the matching predicate signatures, with fact counts and examples, and the matching rules and facts, capped per keyword. A keyword is accepted only if each of its words occurs in the question, in a predicate name or in an earlier search result. The query writer makes at most three search calls, each batched over all the questions of a job.

Logic. Under `facts`, the knowledge base contains ground facts only, and any composition must be written into the query. Under `horn`, it also contains the Horn rules the corpus encoder defined, such as inverses, transitive closures and shorthand relations, which the query writer may call.

Models. Both roles, and both baselines, are played by Claude Opus 5.5. To test whether the conclusions hold for a smaller model, we repeat the full experiment with Claude Haiku 5.5 in every role, re-encoding every knowledge base. Since re-encoding a tenfold corpus for all 84 cells would multiply the cost by ten, on the tenfold corpora we evaluate one cell per grounding with both models, namely the cell with the highest mean F1 over the five corpora at their original size and over both models: the n-ary semantics with `horn` for `kb`, the n-ary semantics with `facts` for `decl`, and the modal semantics with `facts` for `search`; there, RAG@3 retrieves from the enlarged corpus, while the closed-book baseline, which does not see the corpus, is unchanged. For factuality, we evaluate 15 cells: for each grounding, the five with the highest mean F1 over the five corpora at their original size and over both models. The FACTS Grounding judge is Claude Opus 5.5 for every system.

4.5 Learnability of the Corpus and Query Translation Tasks

Model and roles. We train Qwen3-4B-Instruct (Yang et al., 2025a) as two separate instances, one as the corpus encoder and one as the query writer, each with its own LoRA adapter (Hu et al., 2021) of rank 32 ($\alpha = 64$). All runs use the modal semantics with the `search` grounding and the `horn` logic, on the tenfold corpora, so that the query writer must find what it needs in a knowledge base it cannot read.

Training. Each adapter is trained in two stages, supervised fine-tuning followed by reinforcement learning with GRPO (Shao et al., 2024), in the order: corpus encoder SFT, corpus encoder RL, query writer SFT, query writer RL. The SFT data come from Claude Haiku 5.5 as a teacher: its encodings of the training passages, and its search dialogues for the training questions whose final clause binds a correct answer (up to five attempts per question). Both SFT stages run three epochs at learning rate 10^{-4}.

In each RL step, we sample 16 passages for the corpus encoder, half of them supporting passages of training questions and half distractors, or 16 training questions for the query writer, and draw 8 outputs for each. Given an encoding e and a clause c, the reward is computed as

$$r_E(e) = \overline{\mathrm{F1}}(e) + 0.3\,\mathrm{cov}(e) - 0.5\,\mathrm{inv}(e) - 0.5\,\mathrm{uns}(e) - 0.3\,\mathbb{I}[e \text{ is empty}], \tag{5}$$

$$r_Q(c) = \mathrm{F1}(c) - 0.5\,\mathbb{I}[c \text{ states a gold answer as a constant}] - 0.1\,\mathbb{I}[\text{no clause}], \tag{6}$$

where $\overline{\mathrm{F1}}(e)$ is the mean F1 obtained when e replaces the teacher's encoding of its passage and the reference clause of each training question that uses the passage is executed against the result, $\mathrm{cov}(e)$ is the fraction of the passage's names and numbers that appear in the facts, $\mathrm{inv}(e)$ is the fraction of output lines that do not parse, and $\mathrm{uns}(e)$ is the fraction of facts that a judge model (Claude Haiku 5.5) marks as unsupported by the passage. RL runs for 150 corpus encoder steps and 120 query writer steps at learning rate 10^{-5}, and the query writer may search at most twice before writing its clause.

Splits. We train one independent run per dataset (WikiTennis, MuSiQue, 2WikiMultiHopQA and MoreHopQA), holding out 40% of its questions for testing, so that no test question shares a reasoning step with a training question. FanOutQA has too few questions to split, since only 31 of its 310 development questions remain after filtering and fitting the corpus to the shared size, so no run trains on it. We evaluate the base model, and each run after each of its four training stages, on the test questions of the run's own dataset, on the questions of the other datasets, and on the three factuality benchmarks. The test questions of the run's own dataset measure whether the models answer new questions about a corpus they were trained on, which is the specialized use case: the corpus is known in advance and encoded once, so fitting the models closely to it is acceptable. The questions of the other datasets measure whether the models also work on corpora they did not see in training, which is the general use case. The factuality benchmarks, none of which is used in training, measure whether training for question answering also changes how faithfully the models answer from the corpus. As baselines, the base model answers closed-book and with RAG@3.

4.6 Learning the Semantics

Representation and starting points. All runs use the `search` grounding and the `horn` logic, and the query writer may make at most four search calls. The loop starts from the five semantics with the highest mean F1 in the comparison of representations (modal, property graph, event calculus, triple and reified RDF), and from an empty description. The examples in the starting descriptions use invented entities, so that no fact from the corpora reaches the corpus encoder through them.

Models. In the main runs, the corpus encoder and the query writer are the same model, Claude Opus, Sonnet or Haiku 5.5, and Claude Opus 5.5 rewrites the description. Each run performs two rounds of rewriting, and the held-out sets are evaluated with the starting description and with the description that scores best on the training questions. To isolate the effect of the rewriter, we repeat the six runs with Claude Haiku 5.5 as corpus encoder and query writer and with Claude Opus, Sonnet or Haiku 5.5 as the rewriter. Finally, to see whether more rounds help, we run one loop for fifteen rounds, with Claude Haiku 5.5 as corpus encoder and query writer, the property graph as starting point and Claude Opus 5.5 as the rewriter, and evaluate the held-out sets after rounds 2, 5, 10 and 15.

5 Comparing Knowledge Representations

5.1 Question Answering

Figure 2 and Figure 3 place every representation in the plane spanned by tractability and expressiveness, so that cells on the same dotted curve share an F1 score. With Claude Opus 5.5 as corpus encoder and query writer, 53 of the 84 cells score above the closed-book baseline when the five datasets are pooled (0.568), but none reaches RAG@3 (0.827), and only one cell surpasses it on a single dataset, namely skolem-kb-horn on MoreHopQA (0.862 against 0.858). The strongest representations are nary-kb-facts (0.704), modal-kb-horn (0.698), and interval-kb-facts and dl-kb-facts (both 0.694), whereas the weakest are nary-decl-horn (0.419), frame-decl-facts (0.413), and neodav-decl-horn (0.381). Within each dataset, the cells occupy a narrow band of tractability and are separated mainly by expressiveness, which indicates that once a query writer produces a clause that binds, its answer is about equally correct regardless of the representation, and that representations differ chiefly in how often the query binds at all.

Of the three dimensions, grounding separates the cells most clearly. Averaged over all semantics and logics, cells grounded in a fixed knowledge base reach an F1 of 0.666 (expressiveness 0.906, tractability 0.722), cells grounded by search reach 0.633 (0.879, 0.714), and cells grounded in declared predicates reach only 0.462 (0.668, 0.670), so the deficit of declared grounding lies almost entirely in expressiveness. The gap is widest on MuSiQue, where declared grounding binds on 21% of the questions against 72% for the knowledge base and 74% for search, and on FanOutQA, where declared grounding additionally loses tractability (0.35 against 0.65 for the knowledge base). The logic dimension has almost no effect, with facts and Horn clauses at 0.591 and 0.584 and their points intermixed in every panel. The semantic dimension matters less than grounding, since the means of the fourteen semantics lie between 0.546 (neodav) and 0.615 (triple), while the knowledge base cells alone span 0.495 to 0.704.

The same ordering holds for Claude Haiku 5.5, but at a lower level and with a much wider spread in expressiveness. Pooled over the datasets, 48 cells score above the closed-book baseline (0.325) and none reaches RAG@3 (0.689), although skolem-kb-facts surpasses RAG@3 on MoreHopQA (0.825 against 0.793). The best cells are nary-kb-horn (0.556), triple-kb-facts (0.537), and skolem-kb-facts (0.527), while frame-decl-facts, evcalc-decl-facts, and cnl-decl-facts score between 0.060 and 0.071. Grounding again dominates, with means of 0.449 for the knowledge base, 0.352 for search, and 0.130 for declared predicates, whose expressiveness falls to 0.218 and to 0.03 on MuSiQue. The logic dimension remains inert (0.311 against 0.310), and the semantic means range from 0.217 (amr) to 0.370 (modal). MuSiQue and FanOutQA are the hardest datasets for this model, with only 13 and 12 cells above the closed-book baseline, respectively; on FanOutQA, whose questions require lists of answers, tractability stays below 0.40 for every grounding even when expressiveness is high. Across both models, the representation therefore matters most through the

choice of grounding, and a fixed knowledge base or search with a capable model recovers much of the gap between closed-book answering and retrieval without closing it.

Scaling the Corpora

Table 2 and Table 3 compare the best knowledge base, declared, and search cells (nary-kb-horn, nary-decl-facts, and modal-search-facts) on the original corpora and on the tenfold corpora, to which 207 distractor passages were added. For Claude Opus 5.5 on single-query questions, the F1 of RAG@3 falls from 0.827 to 0.748, that of the knowledge base cell from 0.686 to 0.596, and that of the declared cell from 0.481 to 0.371, whereas the search cell loses least, falling from 0.676 to 0.641. In every case the loss comes mainly from expressiveness, while tractability stays nearly unchanged; the knowledge base cell, for instance, binds on 91.9% of the questions on the original corpora and on 83.8% on the tenfold corpora, and on 2WikiMultiHopQA its expressiveness drops from 0.938 to 0.615. The larger corpus thus makes it harder for the query writer to find the predicates and constants that answer a question, but does not make the answers it finds less correct. Multi-query questions degrade less for every arm, with RAG@3 falling from 0.819 to 0.806, the knowledge base cell from 0.722 to 0.664, the declared cell from 0.638 to 0.556, and the search cell from 0.745 to 0.708.

Claude Haiku 5.5 is more sensitive to corpus size. On single-query questions, its knowledge base cell falls from 0.556 to 0.364 as its expressiveness drops from 0.870 to 0.575, its search cell falls from 0.452 to 0.338, and its declared cell halves from 0.199 to 0.095, while RAG@3 loses only 0.026 (0.689 to 0.663). Multi-query questions again show smaller losses (0.559 to 0.463 for the knowledge base, 0.527 to 0.471 for search, and 0.253 to 0.191 for declared predicates). Broken down by the number of hops (Table 12 and Table 13) and of sub-queries (Table 14 and Table 15), the degradation of Claude Opus 5.5 is spread evenly across difficulty levels, whereas the search cell of Claude Haiku 5.5 collapses on four-hop questions (0.321 to 0.104) because its tractability falls from 0.806 to 0.278. A capable model with search grounding therefore scales best among the representations, but no representation scales as well as retrieval for the weaker model.

5.2 Factuality

Table 4 and Table 5 report the five best cells of each grounding on FaithEval, FEVER, and FACTS Grounding, and Table 6 and Table 7 divide the FaithEval and FEVER items of each arm into correct and incorrect answers and correct and incorrect abstentions.

With Claude Opus 5.5, every cell surpasses the closed-book baseline on FaithEval (0.641 to 0.795 against 0.231) and, like RAG@3, declines all nine unanswerable questions. The remaining gap to RAG@3 (0.962) therefore lies in correct answers that the cells decline rather than in wrong answers, which stay at or below 7.7% of the items for every cell except `triple-kb-facts` (14.7%).

On FEVER, the cells answer fewer claims than RAG@3 but answer them more accurately, with tractability between 0.889 and 0.963 against 0.904 and wrong answers on 2.2 to 4.4% of the claims against 6.7%. The knowledge base cells (0.830 to 0.881) and search cells (0.837 to 0.867) thus stay close to RAG@3 (0.896), whereas the declared cells (0.541 to 0.763) decline 20.0 to 43.0% of the claims that the passages support or refute.

FACTS Grounding favors the symbolic route most clearly, since six cells exceed RAG@3 (0.737) and two match it. Four of the five search cells are among them, with `modal-search-facts` at 0.842 and `modal-search-horn` and `nary-search-facts` at 0.789, alongside `triple-kb-facts` (0.895) and `skolem-kb-horn` (0.842), while the declared cells, which bind less often, score at most 0.632.

Claude Haiku 5.5 follows the same pattern at a lower level. On FaithEval, the knowledge base cells (0.686 to 0.788) and search cells (0.577 to 0.660) surpass the closed-book baseline (0.192), and the knowledge base cells answer wrongly less often than RAG@3 (1.9 to 9.6% of the items against 15.4%), whereas the declared cells (0.385 to 0.577) decline 42.3 to 61.5% of the items they should answer. On FEVER, every knowledge base and search cell answers more accurately than RAG@3 (tractability 0.909 to 0.951 against 0.905) and exceeds the closed-book baseline (0.733 to 0.830 against 0.607), while the declared cells fall below it (0.504

to 0.578) because they decline 40.7 to 48.9% of the claims. On FACTS Grounding, no cell reaches RAG@3 (0.737), the best being `triple-kb-horn` (0.684) and `interval-search-facts` (0.632).

Although no representation reaches RAG@3 on FaithEval or FEVER, the representations trade coverage for reliability: every answer is derived from explicit facts that can be traced back to the corpus, and on FEVER these answers are correct more often than those of RAG@3 with both models. Search is the most consistent grounding across benchmarks and models. With Claude Opus 5.5, `modal-search-facts` shares the best FaithEval score (0.795) with the two Skolem knowledge base cells, the search cells stay within 0.06 of RAG@3 on FEVER, and four of them match or exceed RAG@3 on FACTS Grounding; with Claude Haiku 5.5, `nary-search-facts` reaches the highest FEVER score of all cells (0.830). Its losses come from declining correct answers rather than from stating wrong ones, which makes the coverage of the knowledge base, rather than caution, the target for future work.

6 Learnability of the Corpus and Query Translation Tasks

6.1 Question Answering

Reinforcement learning raises the reward of both roles in every run (Figure 4). The corpus encoder reward grows steadily, for instance from 0.18 to 0.29 on WikiTennis and from 0.22 to 0.38 on MoreHopQA, and the query writer reward grows from about zero to 0.41 on WikiTennis and to about 0.3 on 2WikiMultiHopQA and MoreHopQA, but only to 0.11 on MuSiQue.

On the single-query test questions (Figure 5), the mean F1 of the four runs on their own datasets rises from 0.025 for the base model to 0.172 after the last stage, and most of this gain comes from query writer RL. The runs trained on WikiTennis, 2WikiMultiHopQA and MoreHopQA reach 0.285, 0.256 and 0.121, whereas the run trained on MuSiQue ends below the base model (0.024 against 0.073), since its tractability collapses from 0.38 to 0.04. The gains are driven mainly by expressiveness, which rises by 0.12 to 0.43, and they are largely not specific to the training dataset: the run trained on WikiTennis reaches 0.317 on MuSiQue and 0.214 on 2WikiMultiHopQA, and the best run not trained on a dataset averages 0.204, above in-domain training. FanOutQA, on which no run trains, stays at or below 0.029.

Multi-query questions show the same trend (Figure 6): the mean test F1 rises from 0.074 to 0.210, and every run gains between 0.074 and 0.183, again mostly through expressiveness. The trained pipeline thus clearly surpasses the base model answering closed-book (0.059 on single-query and 0.100 on multi-query questions), but stays at or below the base model with RAG@3 (0.204 and 0.411).

6.2 Factuality

Training for question answering also improves factuality, although none of the factuality benchmarks is used in training (Figure 7). FaithEval and FEVER are scored as for the frontier models, with Claude Haiku 5.5 checking the answers. After the last stage, every run improves over the base model on all three benchmarks, by 2.6 to 12.2 percentage points on FaithEval, 3.7 to 20.0 on FEVER, and 5.3 to 10.5 on FACTS Grounding, and the largest FEVER gains come from query writer RL in the runs trained on MuSiQue, 2WikiMultiHopQA and MoreHopQA. The final checkpoints answer only 20 to 38% of the FEVER claims, but their tractability is between 0.98 and 1.00, so the trained models remain faithful: they rarely answer without support in the knowledge base.

7 Learning the Semantics

7.1 The Pipeline on the Category-Diverse Question Set

Table 8 compares the pipeline, before any rewriting and averaged over the six starting semantics, with the closed-book and RAG@3 baselines of each model on the category-diverse question set. On the 705 held-out questions, the pipeline exceeds RAG@3 with all three models: 0.968 against 0.861 with Claude Opus 5.5, 0.926 against 0.889 with Claude Sonnet 5.5, and 0.825 against 0.811 with Claude Haiku 5.5, while the

closed-book baselines reach only 0.791, 0.713 and 0.399. Its advantage lies in expressiveness, since the query binds an answer on 99.0%, 98.2% and 91.5% of the held-out questions against 90.7%, 93.2% and 86.5% for RAG@3, while its tractability is comparable.

The advantage depends on the kind of question. On held-out questions over the training corpora, the pipeline matches RAG@3 with Claude Opus 5.5 (0.950 against 0.951) but stays below it with Claude Sonnet 5.5 (0.891 against 0.941) and Claude Haiku 5.5 (0.786 against 0.900). On questions that combine two categories never seen together in training, and on MoreHopQA, the pipeline is ahead with every model, most clearly on the combinations with Claude Haiku 5.5 (0.775 against 0.606) and on MoreHopQA with Claude Opus 5.5 (0.977 against 0.827). The combinations pair superlatives with the order of events and inverse relations with succession, which a knowledge base answers by executing a query over many facts, whereas retrieval has to find and read every passage involved.

7.2 Learning with Each Model

Table 9 and Figure 8 report the runs in which Claude Opus 5.5 rewrites the semantics for each model as corpus encoder and query writer. Rewriting improves the training score in most runs: a rewritten semantics scores best in five of the six runs with Claude Opus 5.5 and Claude Haiku 5.5, and in four of the six with Claude Sonnet 5.5. The mean gain on the training questions grows as the corpus encoder and query writer become weaker, from 0.010 with Claude Opus 5.5, whose starting semantics already score between 0.952 and 0.973, to 0.023 with Claude Sonnet 5.5 and 0.027 with Claude Haiku 5.5 (Table 10). The largest single gain, 0.071, is that of the property graph with Claude Haiku 5.5, which rises from 0.783 to 0.854 over two rounds.

The gains are specific to the questions used for learning, as is expected of a semantics tuned to a known corpus and its questions. On the held-out questions, the learned semantics improve on average by 0.004 with Claude Opus 5.5 and 0.005 with Claude Sonnet 5.5, and decrease by 0.013 with Claude Haiku 5.5. With Claude Opus 5.5, every learned semantics improves on the held-out questions, though by at most 0.008. The largest held-out gains are those of the empty starting point with Claude Sonnet 5.5 (0.911 to 0.939) and of the event calculus with Claude Haiku 5.5 (0.806 to 0.834), whereas the triple semantics with Claude Haiku 5.5 loses 0.107, mostly on MoreHopQA (0.876 to 0.739).

7.3 Effect of the Rewriter

Table 11 and Figure 9 keep Claude Haiku 5.5 as corpus encoder and query writer and vary the model that rewrites the semantics. A stronger rewriter finds improvements more reliably: with Claude Opus 5.5 or Claude Sonnet 5.5 as the rewriter, a rewritten semantics scores best in five of the six runs, with a mean training gain of 0.027, whereas with Claude Haiku 5.5 only two runs improve, by 0.003 and 0.004, and most of its rewrites score below their starting point. The best training score of the whole study, 0.865, is reached from the empty starting point with Claude Sonnet 5.5 as the rewriter.

On the held-out questions, no rewriter improves on average (Table 10), but the best runs do. With Claude Sonnet 5.5 as the rewriter, the reified RDF semantics rises from 0.782 to 0.816, and its F1 on MoreHopQA, which is never encoded during learning, from 0.787 to 0.862. Appendix C shows how this semantics changed, from the starting description to the learned one, and how the knowledge base changed with it. The largest loss, that of the property graph with Claude Sonnet 5.5 as the rewriter (0.804 to 0.647), comes from expressiveness rather than from wrong answers: on MoreHopQA, its query writer binds an answer to only 295 of the 454 questions, while the answers it binds keep a tractability of 0.905.

7.4 More Rounds of Rewriting

Figure 10 extends one run to fifteen rounds, with Claude Haiku 5.5 as corpus encoder and query writer, the property graph as starting semantics and Claude Opus 5.5 as the rewriter. The training score rises from 0.783 to 0.854 in the first two rounds, and no later round exceeds it: the following rounds score between 0.728 and 0.846. The held-out score is highest after round 5 (0.840, against 0.804 for the starting semantics) and lower after rounds 10 and 15 (0.757 and 0.782). Rewriting thus yields most of its gain in its first rounds, and

further rounds do not improve the semantics. In later rounds, the rewriter also dropped rules that earlier rounds had established, which suggests keeping a fixed list of rules that every rewrite must retain.

8 Conclusion

We studied how large language models can build and use a fully symbolic knowledge base for question answering, with a corpus encoder that translates the corpus into Prolog once and a query writer that translates each question into a single executable clause. Across 84 representations, which combine 14 semantics, three groundings and two logics, and across five multi-hop corpora, their multi-query variants and three factuality benchmarks, every answer of the pipeline is derived from explicit facts and rules that can be inspected and traced back to the corpus. We then asked whether the two translators can be trained, and whether the semantics itself can be learned, on a category-diverse question set over six corpora, with Claude Opus, Sonnet and Haiku 5.5.

For question answering, the grounding of the query writer matters far more than the semantics or the logic. Cells whose query writer sees the knowledge base or searches it outperform those that see only declarations, mainly because their queries bind more often, while facts and Horn rules perform almost alike. With Claude Opus 5.5, 53 of the 84 cells surpass the closed-book baseline, and with Claude Haiku 5.5, 48 do, but no cell reaches RAG@3 when the benchmark datasets are pooled. On the category-diverse question set, by contrast, the pipeline exceeds RAG@3 on the held-out questions with all three models, chiefly on questions that combine categories never seen together and on a corpus never used for learning, while on new questions over the training corpora it matches RAG@3 only with Claude Opus 5.5. On corpora ten times larger, the search grounding loses least with Claude Opus 5.5, whereas every representation degrades more than retrieval with Claude Haiku 5.5.

For factuality, the representations trade coverage for reliability. Once each answer must be established by the facts behind it, the knowledge base and search cells answer fewer FEVER claims than RAG@3 but answer them more accurately with both models, and with Claude Opus 5.5 they decline every unanswerable FaithEval question. On FACTS Grounding, where the response is written from the bound facts alone, six cells exceed RAG@3 with Claude Opus 5.5, four of them search cells.

Both roles can also be learned by an open-weight model. Trained with supervised fine-tuning and reinforcement learning, Qwen3-4B raises its mean test F1 from 0.025 to 0.172 on single-query questions and from 0.074 to 0.210 on multi-query questions, largely independently of the corpus it was trained on, and it becomes more factual on benchmarks it never saw in training. It remains below its own RAG@3 baseline, however, and part of its gain comes from clauses that skip the intermediate hops.

The semantics itself can be learned as well. When a language model rewrites the semantics from the failures of the pipeline, the training score improves in most runs, and more when the corpus encoder and query writer are weaker, by 0.010 with Claude Opus 5.5 and 0.027 with Claude Haiku 5.5 on average. A strong rewriter matters: Claude Opus and Sonnet 5.5 improve five of six runs, whereas Claude Haiku 5.5 barely improves any. The gains come in the first two rounds and concentrate on the questions used for learning, as befits a semantics tuned to a known corpus, but in the best runs they also reach held-out questions and the unseen corpus, where the reified RDF semantics rises from 0.787 to 0.862.

Several directions remain open. First, the corpus encoder and the query writer could be trained jointly from the end-to-end answer, so that the corpus encoder writes what the query writer can use and the query writer adapts to the predicates as they change, instead of training the query writer on knowledge bases that are already fixed. Second, the semantics could be learned with reinforcement learning or with constitutional methods rather than by free-text rewriting (Bai et al., 2022), rewarding the rewriter on batches of questions it has not seen, or binding every rewrite to a fixed set of principles, and to a list of rules that earlier rounds have established and that later rounds may not drop. Third, the corpus encoder could harvest the implicit knowledge of the corpus, writing down not only what the text states but also what it entails or presupposes, such as general rules, missing links between entities and the negative facts that a refutation needs, similarly to how large language models have been used to mine logical rules from knowledge graphs (Luo et al., 2025). Fourth, the setting could be extended to open-domain question answering, where no corpus is given

in advance. The knowledge base would then be built and extended as questions arrive, by encoding the documents retrieved for each question and keeping their facts for later questions, so that it grows over time into a broad symbolic knowledge base in the spirit of Cyc (Lenat, 1995), written by the corpus encoder from text rather than by hand. Finally, several closed-domain corpora could each be served by their own corpus encoder and query writer, specialized to that corpus, with a router that sends each question to the knowledge bases that can answer it.

More broadly, our results suggest that large language models no longer need to be the place where knowledge is stored and reasoning happens, but can instead act as translators into and out of a symbolic system that does both. Such a system keeps the flexibility of the language model at its boundaries while every fact and every inference step in between remains explicit, executable and open to inspection, correction and verification, and even the representation it reasons over can be tuned from its own failures rather than fixed by hand. We hope that this study opens the way towards neuro-symbolic systems, in which the knowledge representation is chosen for what it allows the reasoning to guarantee, and in which trust in an answer rests on a proof that can be checked rather than on the opaque weights of a model.

References

James F Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11): 832–843, 1983.

Renzo Angles. The property graph database model. In *AMW*, 2018.

Franz Baader, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider. *The description logic handbook*, volume 2. Cambridge university press, Cambridge, 2007.

Jiaxin Bai, Wei Fan, Qi Hu, Qing Zong, Chunyang Li, Hong Ting Tsang, Hongyu Luo, Yauwai Yim, Haoyu Huang, Xiao Zhou, et al. Autoschemakg: Autonomous knowledge graph construction through dynamic schema induction from web-scale corpora. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 20557–20584, 2026.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

Collin F Baker, Charles J Fillmore, and John B Lowe. The berkeley framenet project. In *COLING 1998 Volume 1: The 17th International Conference on Computational Linguistics*, 1998.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. Abstract meaning representation for sembanking. In *Proceedings of the 7th linguistic annotation workshop and interoperability with discourse*, pp. 178–186, 2013.

Alexander Beiser, David Penz, and Nysret Musliu. Intermediate languages matter: Formal choice drives neurosymbolic llm reasoning. *arXiv preprint arXiv:2502.17216*, 2025.

Abdelhadi Belfadel, Maxence Gagnant, Joseph Kattan, and Sana Tmar. Llm-guided ontology-driven knowledge graph construction from unstructured text. *arXiv preprint arXiv:2609.31663*, 2026.

Vaishak Belle and Gary Marcus. The future is neuro-symbolic: Where has it been, and where is it going? In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 40954–40961, 2026.

Leonard Bereska and Efstratios Gavves. Mechanistic interpretability for ai safety–a review. *arXiv preprint arXiv:2404.14082*, 2024.

Ruirui Chen, Weifeng Jiang, Chengwei Qin, Bo Xiong, Kaiwen Wei, Fiona Liausvia, Pei Fang Tan, Ker Yung Chua, Dongkyu Choi, Mukkesh Kumar, et al. Are large language models effective knowledge graph constructors? *arXiv preprint arXiv:2510.11297*, 2025.

Donald Davidson. The logical form of action sentences. 1967.

Aghyad Deeb and Fabien Roger. Do unlearning methods remove information from language model weights? *arXiv preprint arXiv:2410.08827*, 2024.

Bhuwan Dhingra, Jeremy R Cole, Julian Martin Eisenschlos, Daniel Gillick, Jacob Eisenstein, and William W Cohen. Time-aware language models as temporal knowledge bases. *Transactions of the Association for Computational Linguistics*, 10:257–273, 2022.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*, 2024.

Xiaohan Feng, Xixin Wu, and Helen Meng. Ontology-grounded automatic knowledge graph construction by llm under wikidata schema. *arXiv preprint arXiv:2412.20942*, 2024.

Tianyu Gao, Howard Yen, Jiatong Yu, and Danqi Chen. Enabling large language models to generate text with citations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 6465–6488, 2023.

Yichen Gao, Altay Unal, Akshay Rangamani, and Zhihui Zhu. An illusion of unlearning? assessing machine unlearning through internal representations. *arXiv preprint arXiv:2604.08271*, 2026.

Bernal J Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. Hipporag: Neurobiologically inspired long-term memory for large language models. *Advances in neural information processing systems*, 37:59532–59569, 2024.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 6609–6625, 2020.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Ziniu Hu, Yichong Xu, Wenhao Yu, Shuohang Wang, Ziyi Yang, Chenguang Zhu, Kai-Wei Chang, and Yizhou Sun. Empowering language models with knowledge graph reasoning for open-domain question answering. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 9562–9581, 2022.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM transactions on information systems*, 43(2): 1–55, 2025.

Alon Jacovi, Andrew Wang, Chris Alberti, Connie Tao, Jon Lipovetz, Kate Olszewska, Lukas Haas, Michelle Liu, Nate Keating, Adam Bloniarz, et al. The facts grounding leaderboard: Benchmarking llms' ability to ground responses to long-form input. *arXiv preprint arXiv:2501.03200*, 2025.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM computing surveys*, 55(12):1–38, 2023.

Zhijing Jin, Yuen Chen, Fernando Gonzalez Adauto, Jiarui Liu, Jiayi Zhang, Julian Michael, Bernhard Schölkopf, and Mona Diab. Analyzing the role of semantic representations in the era of large language models. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 3781–3798, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.209.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Saiful Haq, Ashutosh Sharma, Thomas Joshi, Hanna Moazam, Heather Miller, et al. Dspy: Compiling declarative language model calls into state-of-the-art pipelines. In *International Conference on Learning Representations*, volume 2024, pp. 54928–54958, 2024.

Robert Kowalski and Marek Sergot. A logic-based calculus of events. *New generation computing*, 4(1):67–95, 1986.

Tobias Kuhn. A survey and classification of controlled natural languages. *Computational linguistics*, 40(1): 121–170, 2014.

Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, et al. Measuring faithfulness in chain-of-thought reasoning. *arXiv preprint arXiv:2307.13702*, 2023.

Angeliki Lazaridou, Adhi Kuncoro, Elena Gribovskaya, Devang Agrawal, Adam Liska, Tayfun Terzi, Mai Gimenez, Cyprien de Masson d'Autume, Tomas Kocisky, Sebastian Ruder, et al. Mind the gap: Assessing temporal generalization in neural language models. *Advances in Neural Information Processing Systems*, 34:29348–29363, 2021.

Douglas B Lenat. Cyc: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11):33–38, 1995.

Hector J Levesque and Ronald J Brachman. *A fundamental tradeoff in knowledge representation and reasoning*. Laboratory for Artificial Intelligence Research, Fairchild, Schlumberger, 1984.

Nelson F Liu, Tianyi Zhang, and Percy Liang. Evaluating verifiability in generative search engines. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 7001–7025, 2023.

Jakub Łucki, Boyi Wei, Yangsibo Huang, Peter Henderson, Florian Tramèr, and Javier Rando. An adversarial perspective on machine unlearning for ai safety. *arXiv preprint arXiv:2409.18025*, 2024.

Haoran Luo, E Haihong, Zichen Tang, Shiyao Peng, Yikai Guo, Wentai Zhang, Chenghao Ma, Guanting Dong, Meina Song, Wei Lin, et al. Chatkbqa: A generate-then-retrieve framework for knowledge base question answering with fine-tuned large language models. In *Findings of the association for computational linguistics: ACL 2024*, pp. 2039–2056, 2024.

Linhao Luo, Jiaxin Ju, Bo Xiong, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. Chatrule: Mining logical rules with large language models for knowledge graph reasoning. In *Pacific-asia conference on knowledge discovery and data mining*, pp. 314–325. Springer, 2025.

Aengus Lynch, Phillip Guo, Aidan Ewart, Stephen Casper, and Dylan Hadfield-Menell. Eight methods to evaluate robust unlearning in llms. *arXiv preprint arXiv:2402.16835*, 2024.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in neural information processing systems*, 36:46534–46594, 2023.

Jiayuan Mao, Chuang Gan, Pushmeet Kohli, Joshua B Tenenbaum, and Jiajun Wu. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. *arXiv preprint arXiv:1904.12584*, 2019.

Yifei Ming, Senthil Purushwalkam, Shrey Pandit, Zixuan Ke, Xuan-Phi Nguyen, Caiming Xiong, and Shafiq Joty. Faitheval: Can your language model stay faithful to context, even if" the moon is made of marshmallows". In *International Conference on Learning Representations*, volume 2025, pp. 29430–29456, 2025.

Theo Olausson, Alex Gu, Ben Lipkin, Cedegao Zhang, Armando Solar-Lezama, Joshua Tenenbaum, and Roger Levy. Linc: A neurosymbolic approach for logical reasoning by combining language models with first-order logic provers. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 5153–5176, 2023.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Wang. Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 3806–3824, 2023.

Pranoy Panda, Ankush Agarwal, Chaitanya Devaguptapu, Manohar Kaul, and Prathosh Ap. HOLMES: Hyper-relational knowledge graphs for multi-hop question answering using LLMs. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13263–13282, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.717.

Vaidehi Ramesh Patil, Peter Hase, and Mohit Bansal. Can sensitive information be deleted from llms? objectives for defending against extraction attacks. In *International Conference on Learning Representations*, volume 2024, pp. 45497–45514, 2024.

Reid Pryzant, Dan Iter, Jerry Li, Yin Lee, Chenguang Zhu, and Michael Zeng. Automatic prompt optimization with "gradient descent" and beam search. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 7957–7968, 2023.

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Foundations and trends® in information retrieval*, 4(1-2):1–174, 2009.

Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature machine intelligence*, 1(5):206–215, 2019.

Julian Schnitzler, Xanh Ho, Jiahao Huang, Florian Boudin, Saku Sugawara, and Akiko Aizawa. Morehopqa: More than multi-hop reasoning. *arXiv preprint arXiv:2406.13397*, 2024.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36: 8634–8652, 2023.

Thoralf Skolem. Logisch-kombinatorische untersuchungen über die erfüllbarkeit oder bewiesbarkeit mathematischer sätze nebst einem theorem über dichte mengen. 1920.

Mariarosaria Taddeo and Luciano Floridi. Solving the symbol grounding problem: a critical review of fifteen years of research. *Journal of Experimental & Theoretical Artificial Intelligence*, 17(4):419–445, 2005.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. Fever: a large-scale dataset for fact extraction and verification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 809–819, 2018.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. musique: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. *Advances in Neural Information Processing Systems*, 36:74952–74965, 2023.

Chao Wan, Albert Gong, Mihir Mishra, Carl-Leander Henneking, Claas Beger, and Kilian Q Weinberger. π-cot: Prolog-initialized chain-of-thought prompting for multi-hop question-answering. *arXiv preprint arXiv:2506.20642*, 2025.

Jingjin Wang and Jiawei Han. PropRAG: Guiding retrieval with beam search over proposition paths. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 6212–6227, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.317.

Xiangyu Wang, Haocheng Yang, Fengxiang Cheng, and Fenrong Liu. Adaptive selection of symbolic languages for improving llm logical reasoning, 2025.

Jan Wielemaker, Tom Schrijvers, Markus Triska, and Torbjörn Lager. Swi-prolog. *Theory and Practice of Logic Programming*, 12(1-2):67–96, 2012.

Chengjin Xu, Muzhi Li, Cehao Yang, Xuhui Jiang, Lumingyuan Tang, Yiyan Qi, and Jian Guo. Context graph, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *International Conference on Learning Representations*, volume 2024, pp. 12028–12068, 2024.

Xiao-Wen Yang, Jie-Jing Shao, Lan-Zhe Guo, Bo-Wen Zhang, Zhi Zhou, Lin-Han Jia, Wang-Zhou Dai, and Yu-Feng Li. Neuro-symbolic artificial intelligence: Towards improving the reasoning abilities of large language models. *arXiv preprint arXiv:2508.13678*, 2025b.

Xi Ye, Qiaochu Chen, Isil Dillig, and Greg Durrett. Satlm: Satisfiability-aided language models using declarative prompting. *Advances in Neural Information Processing Systems*, 36:45548–45580, 2023.

Julien Yu, Mihaela Bornea, Avi Sil, and Katrin Kirchhoff. Graph serializations for reasoning LLMs: A study across knowledge extraction and structural reasoning. In Hung-Yi Lee, Sunipa Dev, Jessy Li, and Desmond Elliott (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2026*, pp. 11779–11794, Budapest, Hungary, October 2026. Association for Computational Linguistics. ISBN 979-8-89176-462-0. doi: 10.18653/2026.findings-emnlp.558.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Pan Lu, Zhi Huang, Carlos Guestrin, and James Zou. Optimizing generative ai by backpropagating language model feedback. *Nature*, 639(8055): 609–616, 2025.

Zhiqiang Zhang, Liqiang Wen, and Wen Zhao. Rule-kbqa: Rule-guided reasoning for complex knowledge base question answering with large language models. In *Proceedings of the 31st international conference on computational linguistics*, pp. 8399–8417, 2025.

Haiyan Zhao, Hanjie Chen, Fan Yang, Ninghao Liu, Huiqi Deng, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, and Mengnan Du. Explainability for large language models: A survey. *ACM Transactions on Intelligent Systems and Technology*, 15(2):1–38, 2024.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. *arXiv preprint arXiv:2211.01910*, 2022.

Andrew Zhu, Alyssa Hwang, Liam Dugan, and Chris Callison-Burch. Fanoutqa: A multi-hop, multi-document question answering benchmark for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 18–37, 2024.

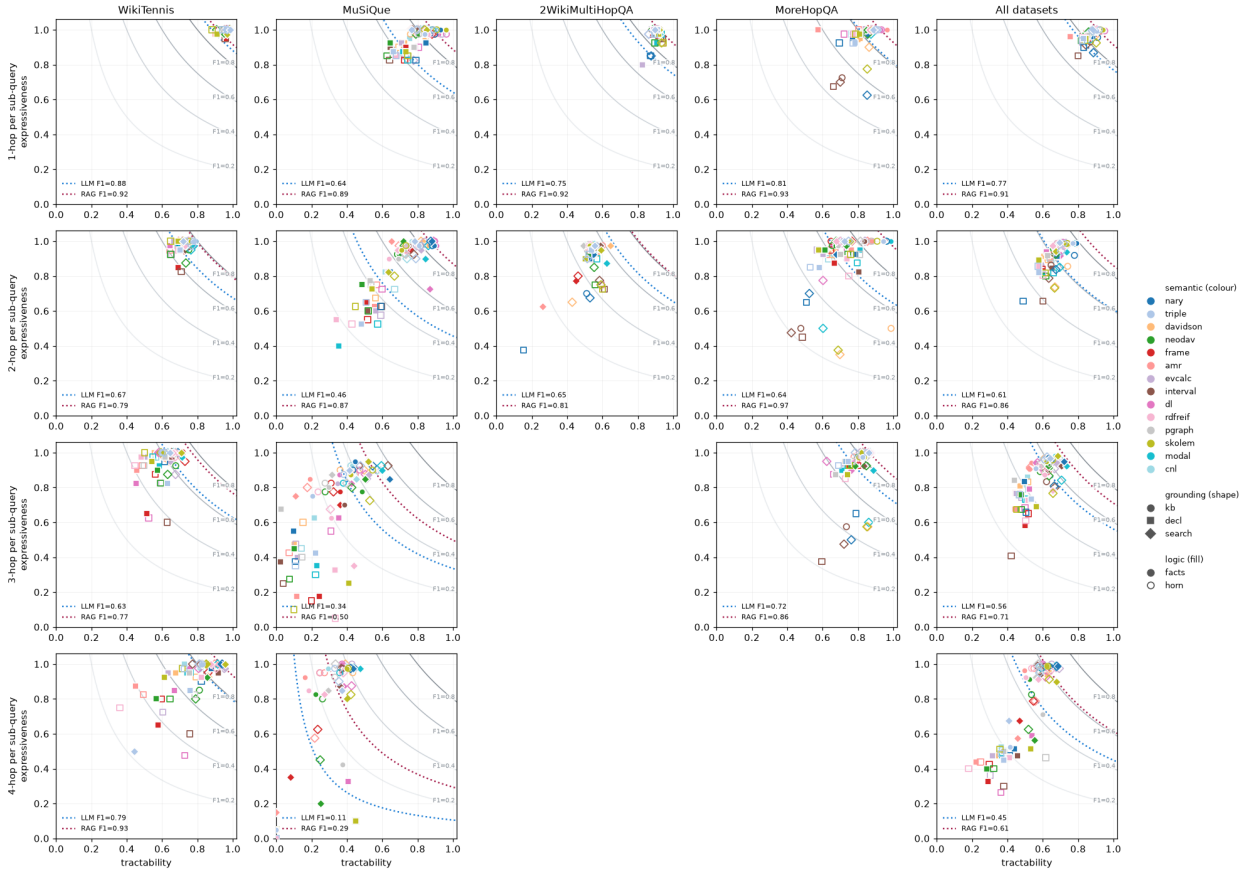


Figure 11: Expressiveness against tractability of every representation with Claude Opus 5.5 on the multi-query questions, by the number of hops per sub-query (rows) and by dataset (columns). Color, marker shape, and fill encode the semantic, the grounding, and the logic, and dotted curves mark the F1 of the closed-book (LLM) and RAG baselines.

A Additional Knowledge Representations Comparison Results

A.1 Question Answering

Multi-Hop Question Answering

Figures 11–18 break the multi-query results down by the number of hops of each sub-query. With Claude Opus 5.5, both baselines degrade as the hops increase, from 0.77 to 0.45 for closed-book answering and from 0.91 to 0.61 for RAG@3 over all datasets, whereas the knowledge base and search cells keep an expressiveness close to one and lose mainly tractability. On four-hop sub-queries, several of them therefore reach or exceed RAG@3, most clearly on MuSiQue, where RAG@3 falls to 0.29. The declared cells instead lose expressiveness as the hops increase. With Claude Haiku 5.5, every grounding loses expressiveness with the number of hops, the declared cells most, and RAG@3 stays above almost every cell. At every number of hops, the facts and Horn cells are intermixed, and the semantics show no consistent order.

Multi-Query Question Answering

Figures 19–26 group the same questions by their number of sub-queries instead. Here neither the baselines nor the cells degrade systematically: with Claude Opus 5.5, closed-book answering stays between 0.61 and 0.68 and RAG@3 between 0.80 and 0.85 from one to four sub-queries, and the knowledge base and search

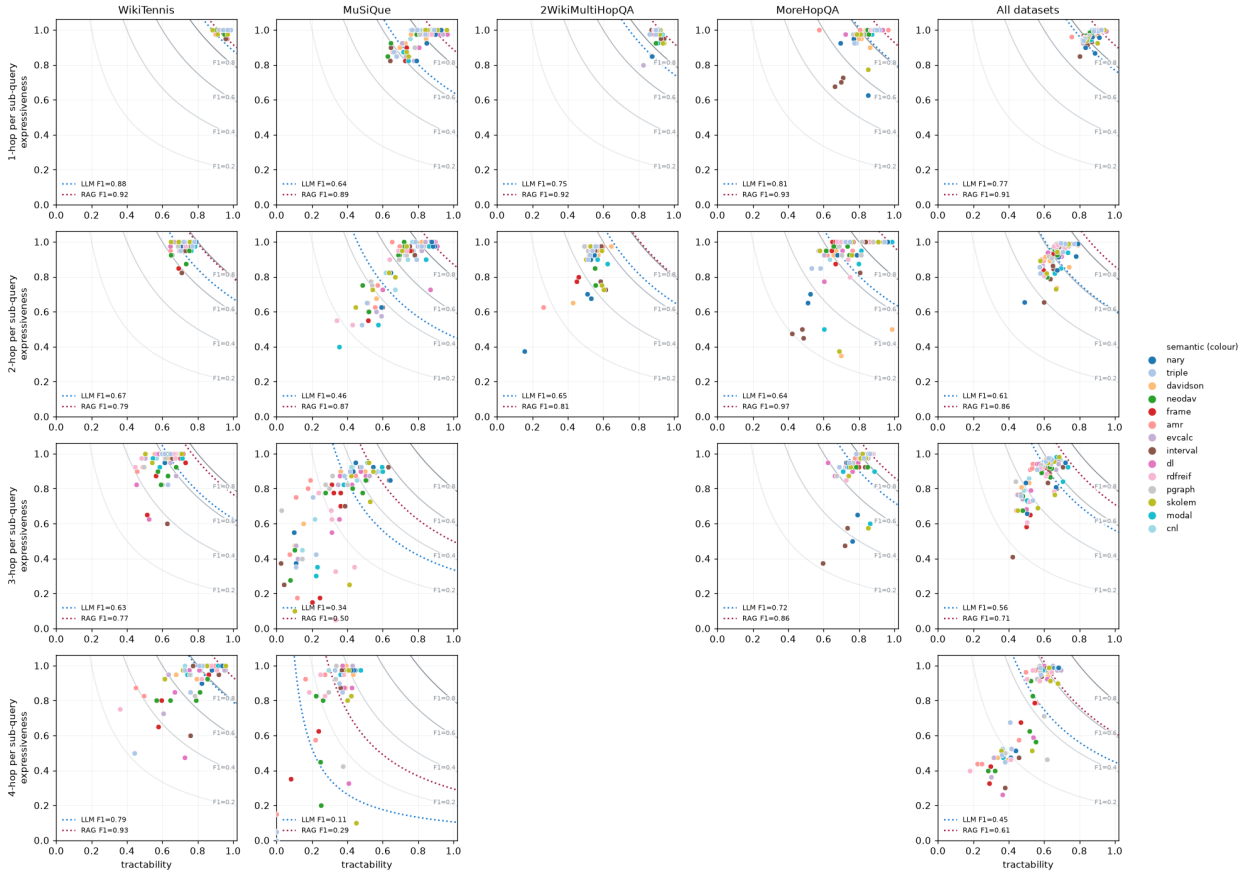


Figure 12: The cells of Figure 11, colored by semantic.

cells remain in the same region of the plane, since each sub-query is answered by its own part of the clause. With Claude Haiku 5.5, the expressiveness of the knowledge base and search cells even rises slightly with the number of sub-queries, because a clause that answers only some of them still binds an answer, while the declared cells remain the least expressive throughout.

A.2 Scaling the Corpora

Multi-Hop Question Answering

Table 12 and Table 13 break the comparison between the original and tenfold corpora down by the number of hops of the single-query questions. With Claude Opus 5.5, the losses are spread over the hop counts rather than concentrated on the longest chains: the knowledge base cell loses 0.151, 0.036 and 0.085 F1 on two, three and four hops, the search cell 0.018, 0.036 and 0.068, and RAG@3 0.109, 0.050 and 0.117. With Claude Haiku 5.5, the losses grow with the hops instead, most sharply for the search cell on four hops (0.321 to 0.104), whose tractability falls from 0.806 to 0.278, and for the knowledge base cell on three hops (0.667 to 0.257).

Multi-Query Question Answering

By the number of sub-queries (Table 14 and Table 15), the losses do not grow with the size of the question for either model. RAG@3 changes by at most 0.03 F1 at every number of sub-queries, the search cell loses between 0.014 and 0.047 with Claude Opus 5.5, and the knowledge base cell loses up to 0.131 with Claude Haiku 5.5, on questions with four sub-queries.

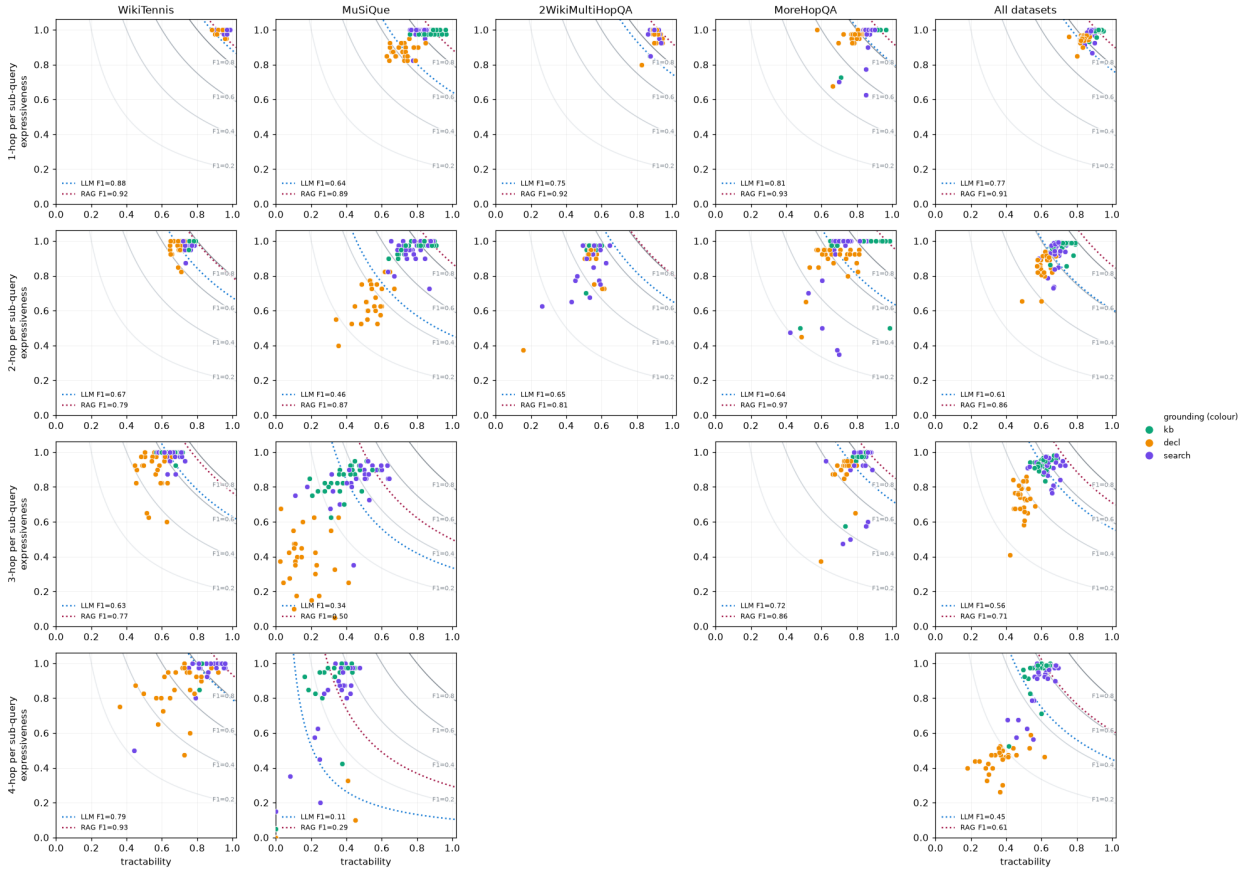


Figure 13: The cells of Figure 11, colored by grounding.

			Closed-book	RAG@3	`nary-kb-horn`			`nary-decl-facts`			`modal-search-facts`		
Hops	n	Corpus	F1	F1	E	T	F1	E	T	F1	E	T	F1
2	189	Original	0.535	0.860	0.927	0.759	0.702	0.677	0.679	0.442	0.953	0.687	0.655
		Tenfold		0.751	0.790	0.706	0.551	0.540	0.613	0.325	0.915	0.697	0.637
3	147	Original	0.671	0.835	0.921	0.767	0.719	0.778	0.709	0.621	0.926	0.765	0.721
		Tenfold		0.785	0.874	0.741	0.683	0.644	0.688	0.531	0.862	0.761	0.685
4	61	Original	0.612	0.842	0.951	0.788	0.763	0.609	0.815	0.527	0.963	0.759	0.740
		Tenfold		0.725	0.907	0.749	0.678	0.491	0.606	0.446	0.887	0.742	0.672

Table 12: Expressiveness (E), tractability (T), and F1 of the baselines and of the best knowledge base, declared, and search cells with Claude Opus 5.5 on the original and tenfold corpora, by number of hops, averaged over the single-query datasets; n is the number of questions pooled.

B Question Categories

Table 16 lists the 23 question categories of the category-diverse question set, in eight groups. Each category is crossed with one to four hops, with single or multiple sub-queries, and with the type of the answer: an entity, a set, a number, a date, yes or no, or unknown. The examples show the kind of question each category contains; the entities in them are placeholders.

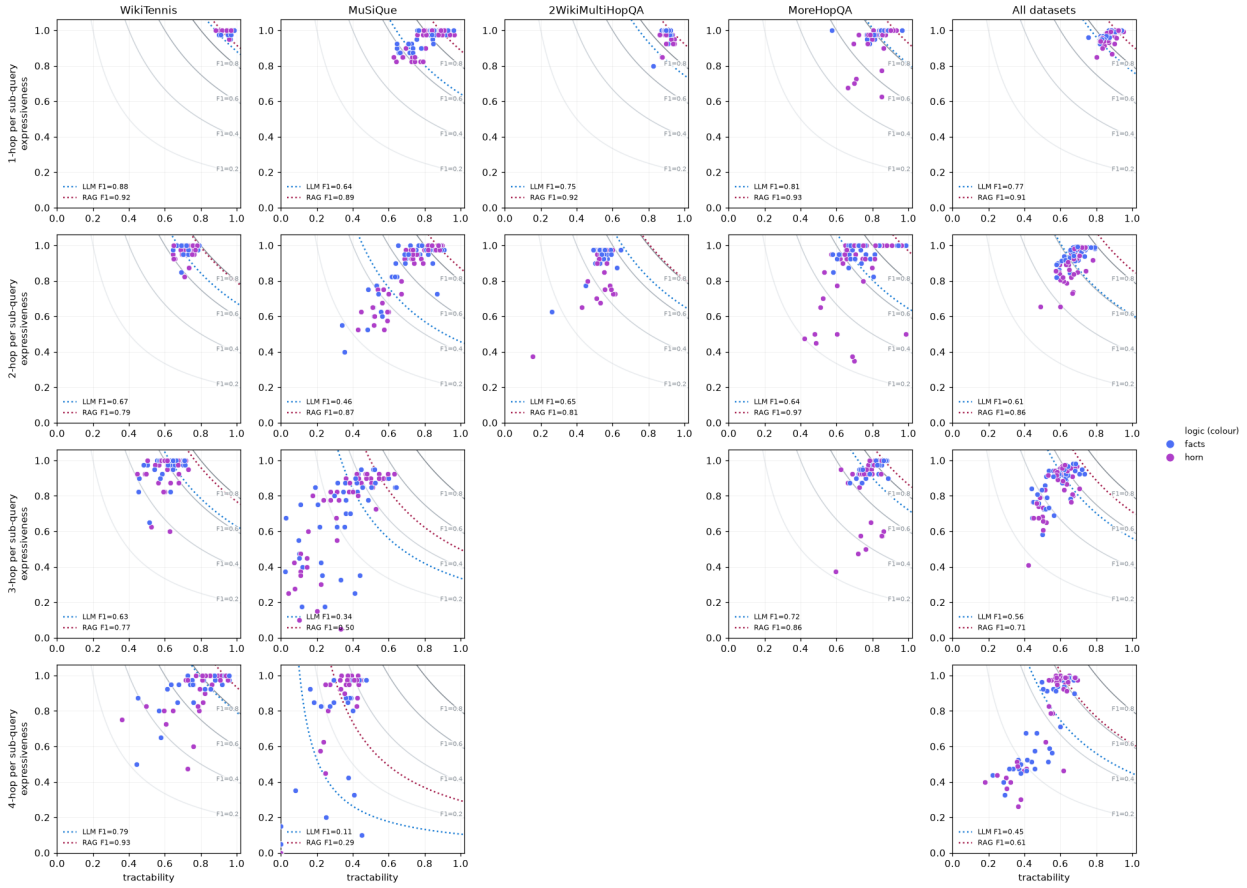


Figure 14: The cells of Figure 11, colored by logic.

			Closed-book	RAG@3	`nary-kb-horn`			`nary-decl-facts`			`modal-search-facts`		
Hops	n	Corpus	F1	F1	E	T	F1	E	T	F1	E	T	F1
2	189	Original	0.327	0.691	0.881	0.576	0.508	0.339	0.590	0.184	0.755	0.637	0.483
		Tenfold		0.679	0.648	0.577	0.385	0.236	0.452	0.111	0.530	0.663	0.339
3	147	Original	0.207	0.750	0.881	0.734	0.667	0.398	0.654	0.368	0.695	0.758	0.569
		Tenfold		0.713	0.616	0.370	0.257	0.287	0.660	0.281	0.586	0.641	0.504
4	61	Original	0.351	0.717	0.745	0.767	0.596	0.097	0.611	0.090	0.390	0.806	0.321
		Tenfold		0.698	0.605	0.537	0.441	0.056	0.333	0.056	0.162	0.278	0.104

Table 13: Expressiveness (E), tractability (T), and F1 of the baselines and of the best knowledge base, declared, and search cells with Claude Haiku 5.5 on the original and tenfold corpora, by number of hops, averaged over the single-query datasets; n is the number of questions pooled.

The procedural category covers the order of steps, their preconditions, the agent and object of each step, branches and effects, conditions and loops stated in prose, as in recipes, voting rules and game rules, and the execution of a described procedure on a given input, such as determining who wins a tie-break from a given sequence of points.

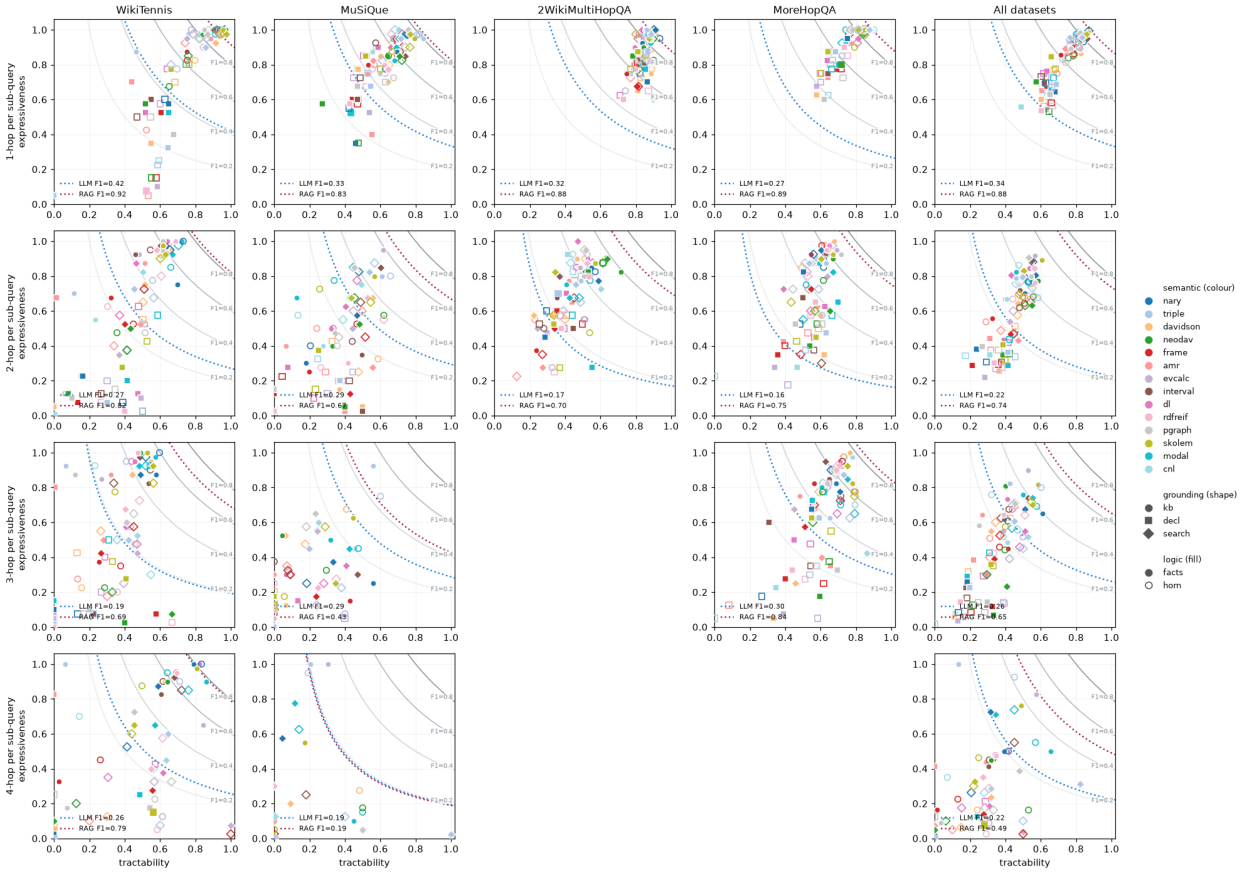


Figure 15: Expressiveness against tractability of every representation with Claude Haiku 5.5 on the multi-query questions, by the number of hops per sub-query (rows) and by dataset (columns), with the same encoding as Figure 11.

			Closed-book	RAG@3	`nary-kb-horn`			`nary-decl-facts`			`modal-search-facts`		
Sub-queries	n	Corpus	F1	F1	E	T	F1	E	T	F1	E	T	F1
1	130	Original	0.612	0.802	0.869	0.802	0.709	0.715	0.831	0.586	0.910	0.791	0.724
		Tenfold		0.793	0.810	0.783	0.636	0.633	0.782	0.495	0.931	0.726	0.681
2	130	Original	0.634	0.825	0.929	0.789	0.734	0.890	0.733	0.665	0.975	0.744	0.727
		Tenfold		0.809	0.875	0.720	0.629	0.823	0.652	0.562	0.965	0.742	0.713
3	130	Original	0.677	0.850	0.950	0.760	0.725	0.925	0.712	0.673	0.981	0.785	0.770
		Tenfold		0.843	0.963	0.766	0.734	0.823	0.690	0.598	0.971	0.746	0.723
4	130	Original	0.630	0.800	0.975	0.737	0.719	0.931	0.661	0.627	0.956	0.793	0.759
		Tenfold		0.781	0.902	0.722	0.659	0.863	0.636	0.568	1.000	0.712	0.712

Table 14: Expressiveness (E), tractability (T), and F1 of the baselines and of the best knowledge base, declared, and search cells with Claude Opus 5.5 on the original and tenfold corpora, by number of sub-queries, averaged over the multi-query datasets; n is the number of questions pooled.

The two category combinations held out from training are B6 with C10, superlatives over the order of events, and A3 with C12, inverse relations over succession.

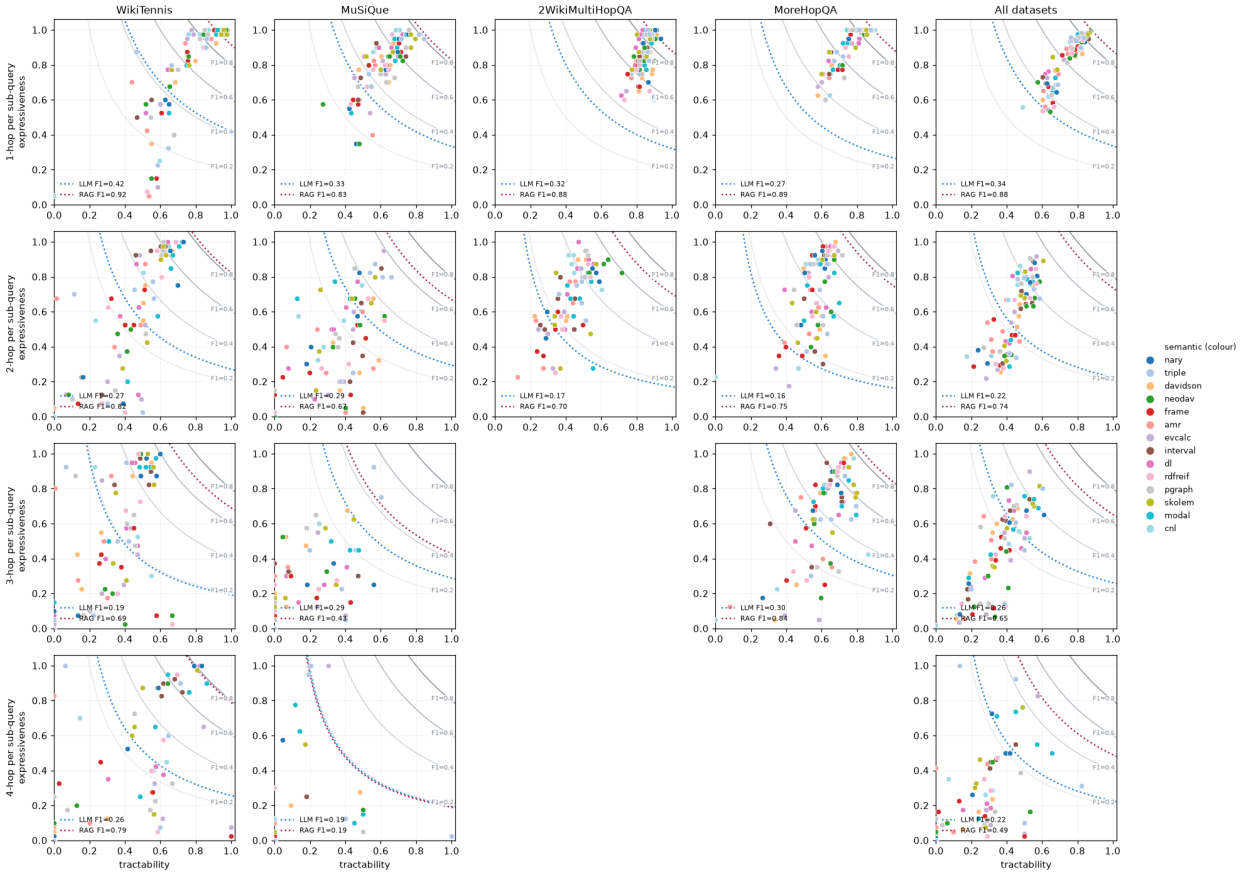


Figure 16: The cells of Figure 15, colored by semantic.

			Closed-book	RAG@3	`nary-kb-horn`			`nary-decl-facts`			`modal-search-facts`		
Sub-queries	n	Corpus	F1	F1	E	T	F1	E	T	F1	E	T	F1
1	130	Original	0.243	0.750	0.742	0.734	0.561	0.348	0.751	0.265	0.738	0.709	0.532
		Tenfold		0.723	0.710	0.760	0.532	0.225	0.817	0.190	0.635	0.760	0.482
2	130	Original	0.309	0.714	0.781	0.650	0.555	0.429	0.639	0.290	0.894	0.617	0.555
		Tenfold		0.729	0.665	0.617	0.435	0.308	0.622	0.206	0.719	0.599	0.441
3	130	Original	0.289	0.795	0.790	0.706	0.586	0.438	0.454	0.219	0.835	0.617	0.529
		Tenfold		0.766	0.738	0.654	0.478	0.346	0.552	0.194	0.756	0.670	0.518
4	130	Original	0.213	0.694	0.760	0.667	0.535	0.502	0.429	0.237	0.800	0.617	0.490
		Tenfold		0.683	0.683	0.577	0.404	0.340	0.505	0.176	0.742	0.587	0.443

Table 15: Expressiveness (E), tractability (T), and F1 of the baselines and of the best knowledge base, declared, and search cells with Claude Haiku 5.5 on the original and tenfold corpora, by number of sub-queries, averaged over the multi-query datasets; n is the number of questions pooled.

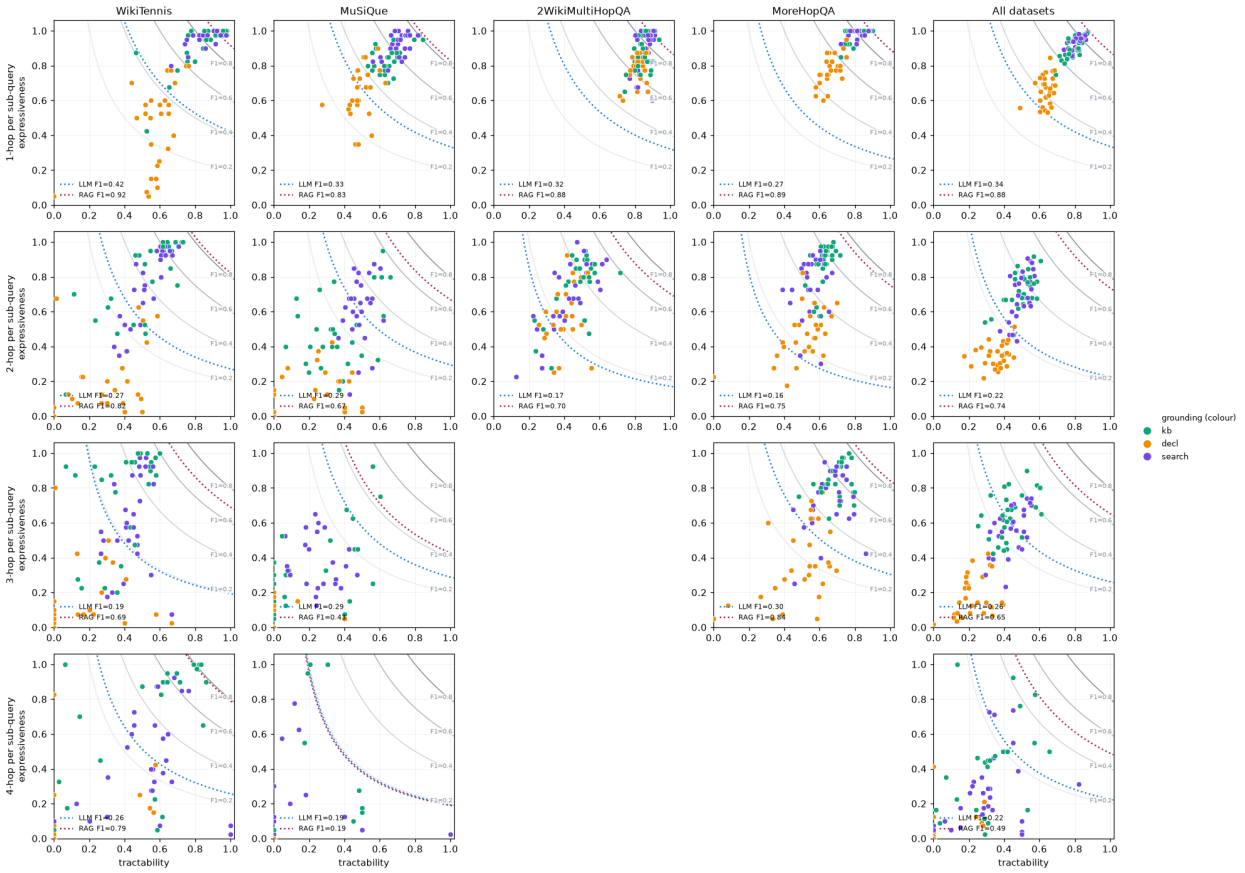


Figure 17: The cells of Figure 15, colored by grounding.

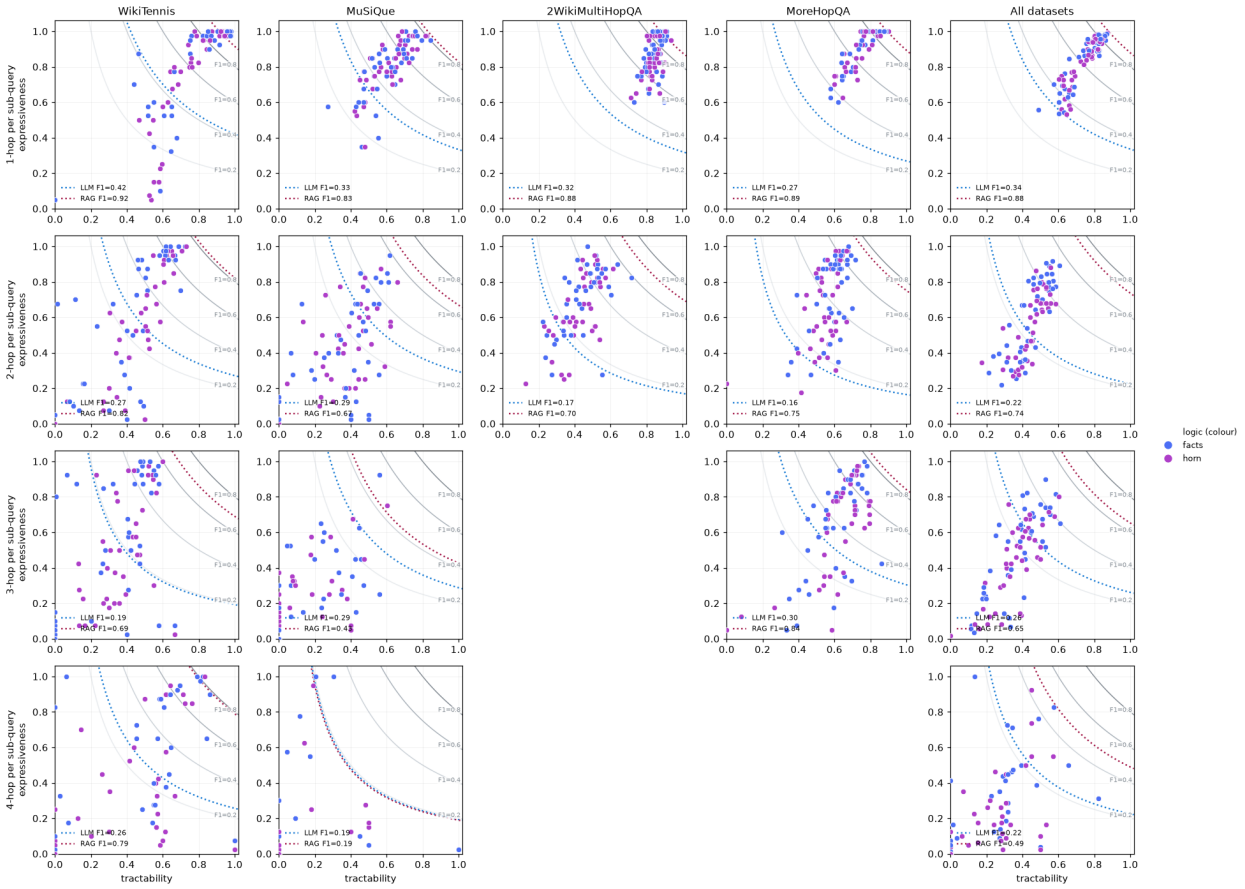


Figure 18: The cells of Figure 15, colored by logic.

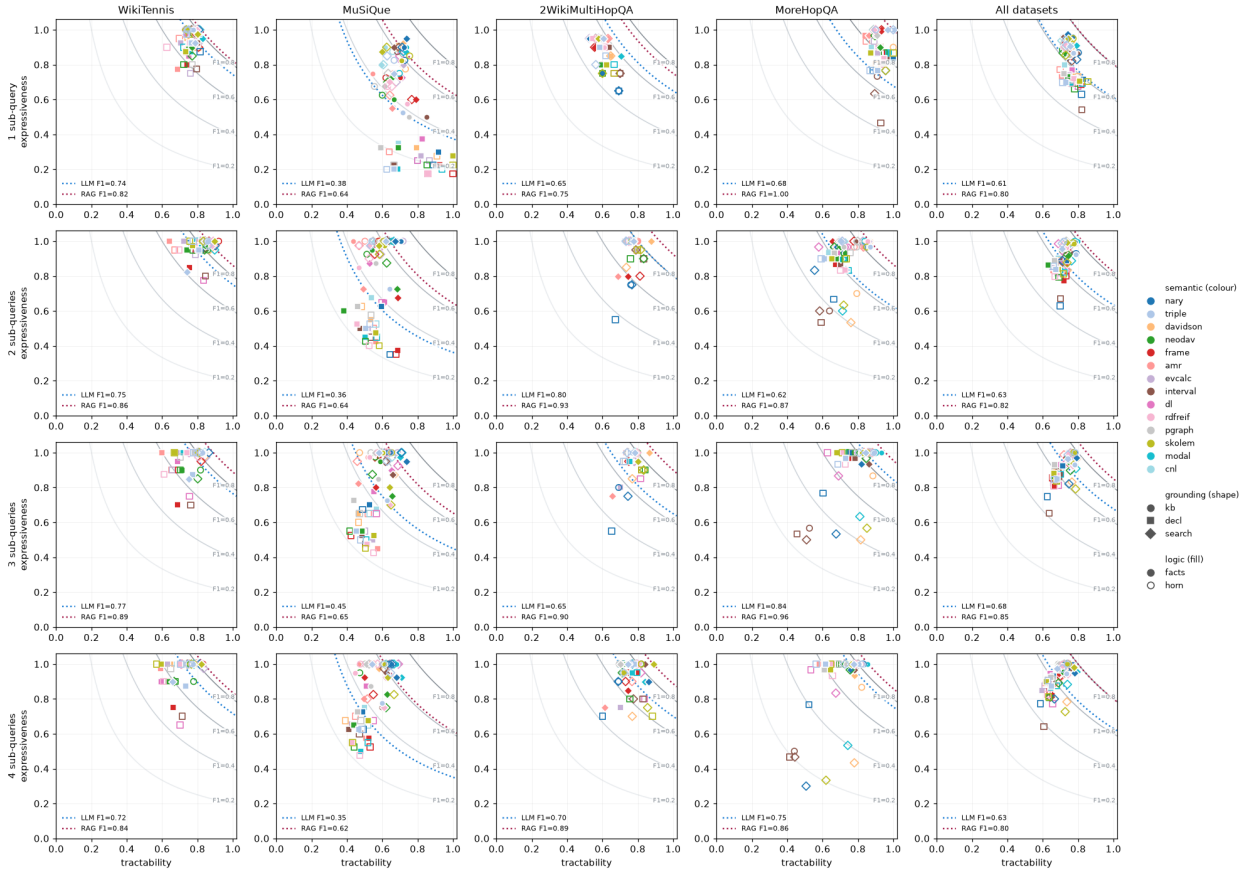


Figure 19: Expressiveness against tractability of every representation with Claude Opus 5.5 on the multi-query questions, by the number of sub-queries (rows) and by dataset (columns), with the same encoding as Figure 11.

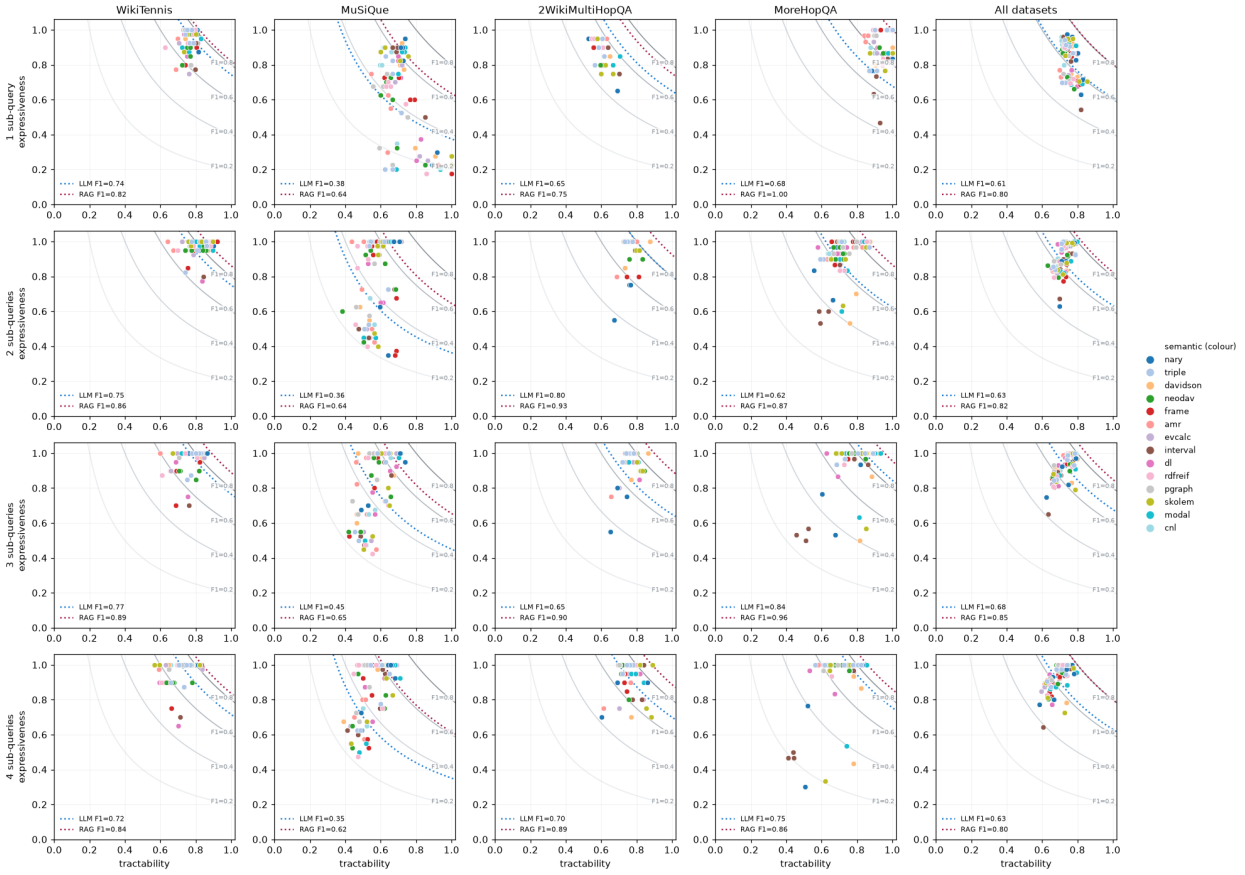


Figure 20: The cells of Figure 19, colored by semantic.

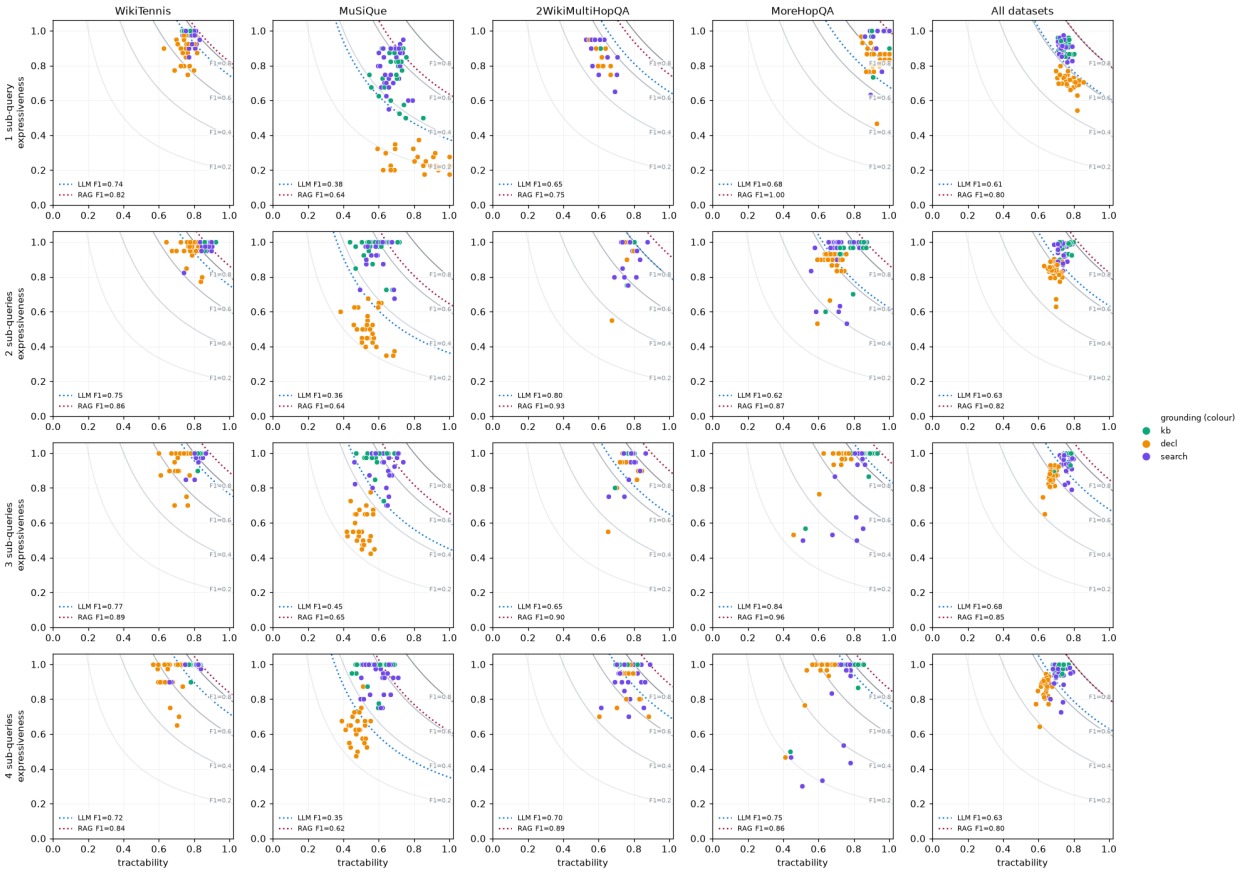


Figure 21: The cells of Figure 19, colored by grounding.

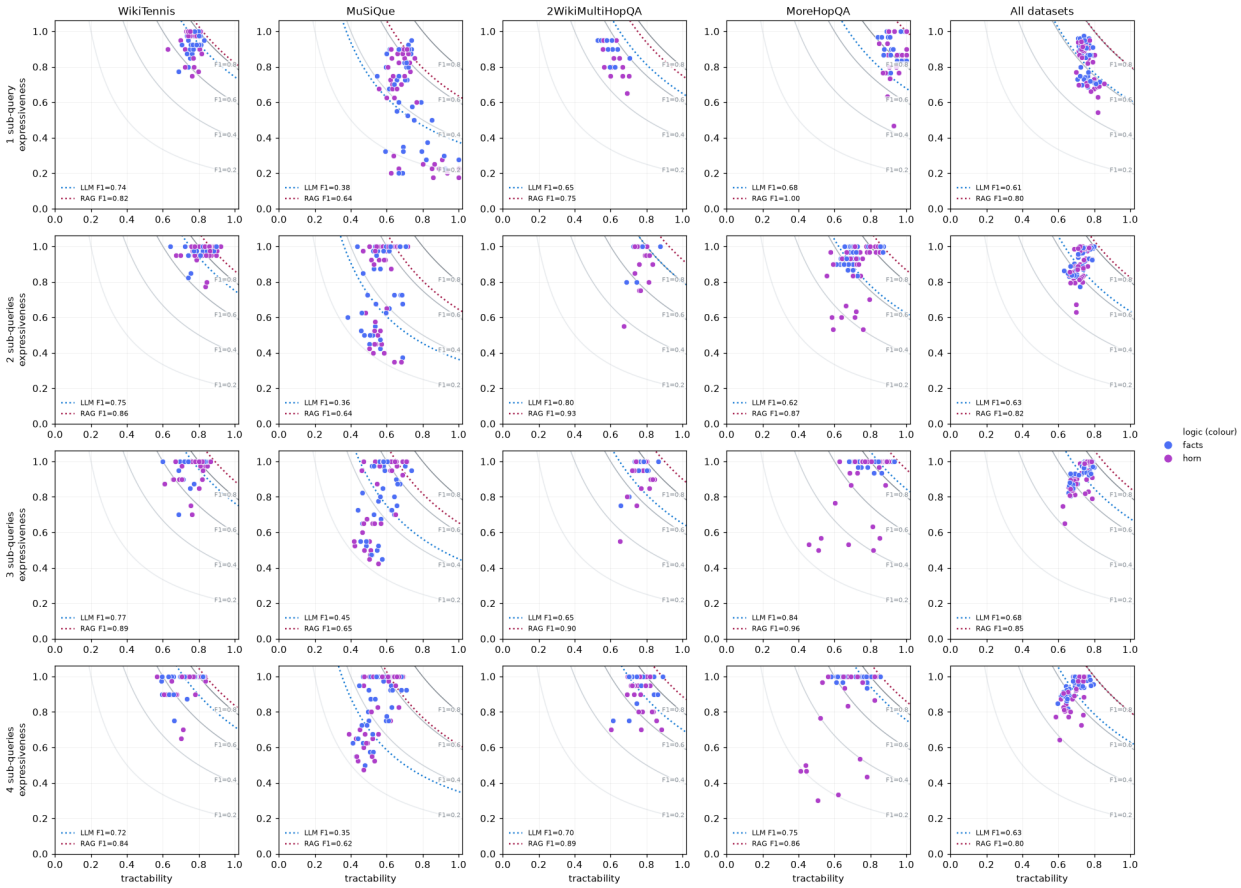


Figure 22: The cells of Figure 19, colored by logic.

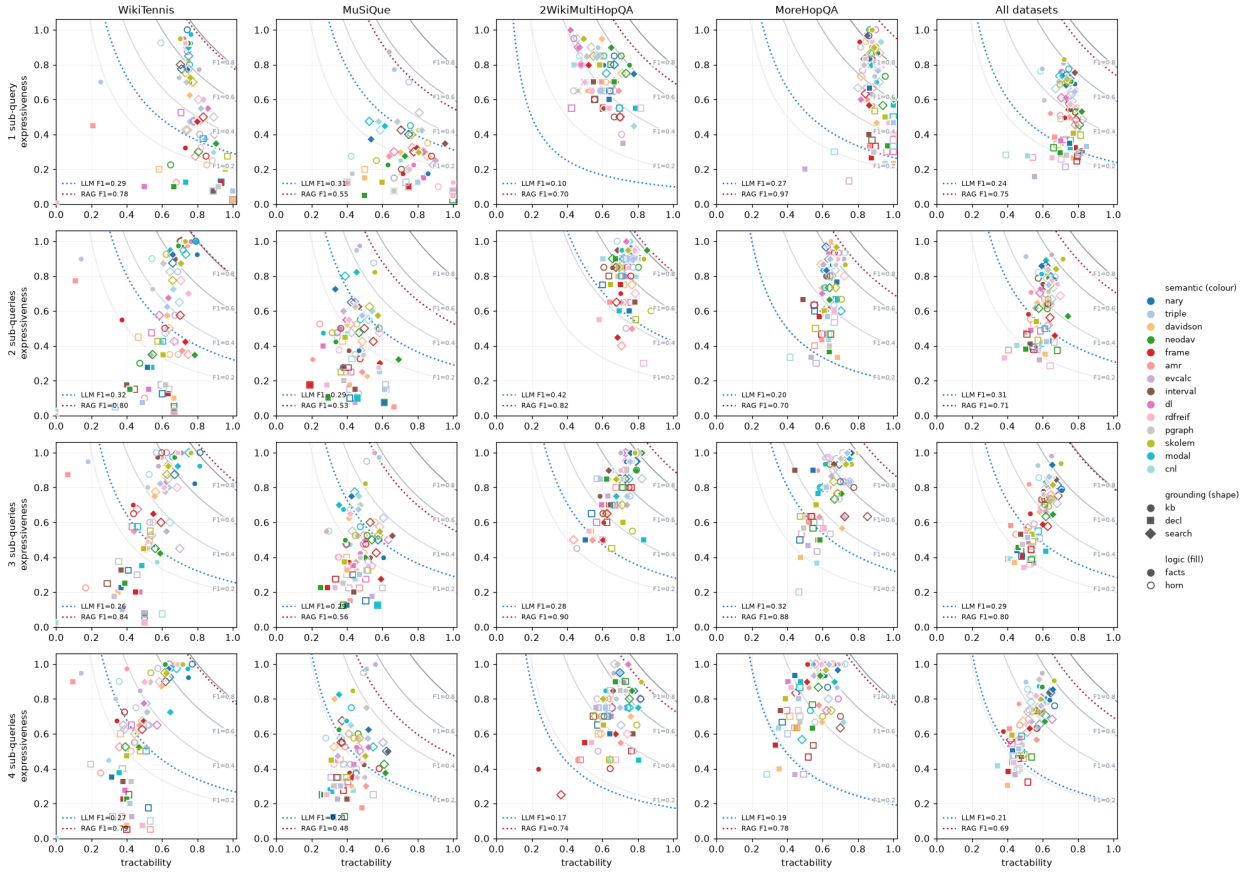


Figure 23: Expressiveness against tractability of every representation with Claude Haiku 5.5 on the multi-query questions, by the number of sub-queries (rows) and by dataset (columns), with the same encoding as Figure 11.

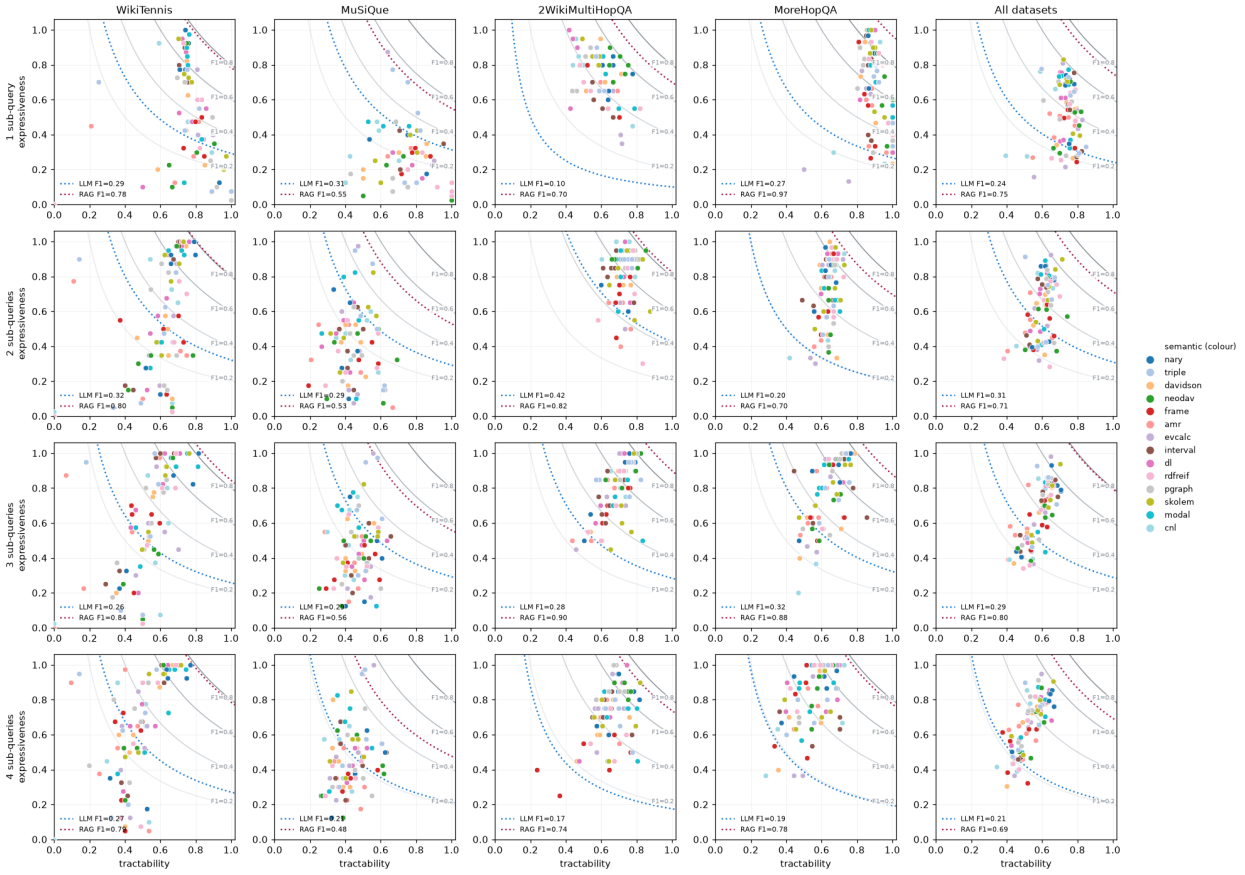


Figure 24: The cells of Figure 23, colored by semantic.

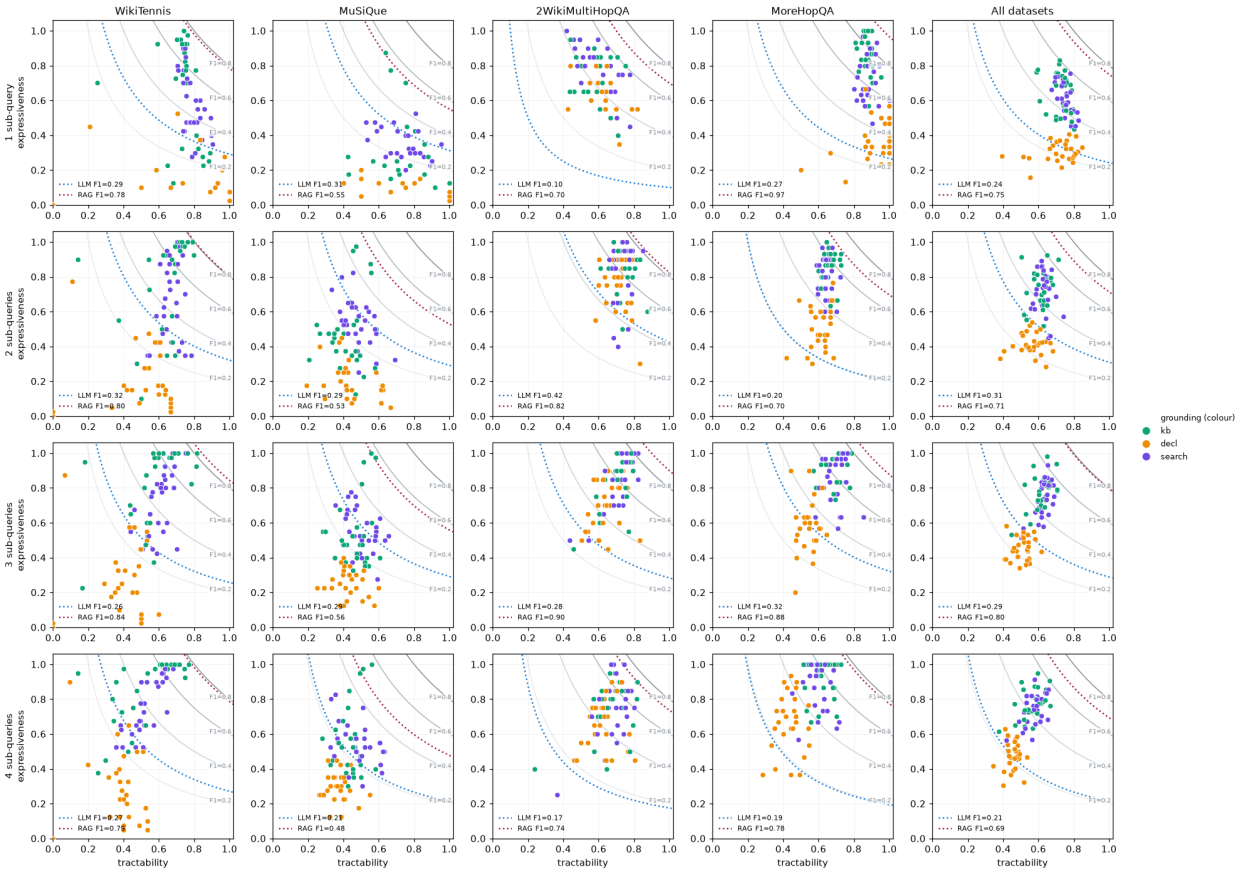


Figure 25: The cells of Figure 23, colored by grounding.

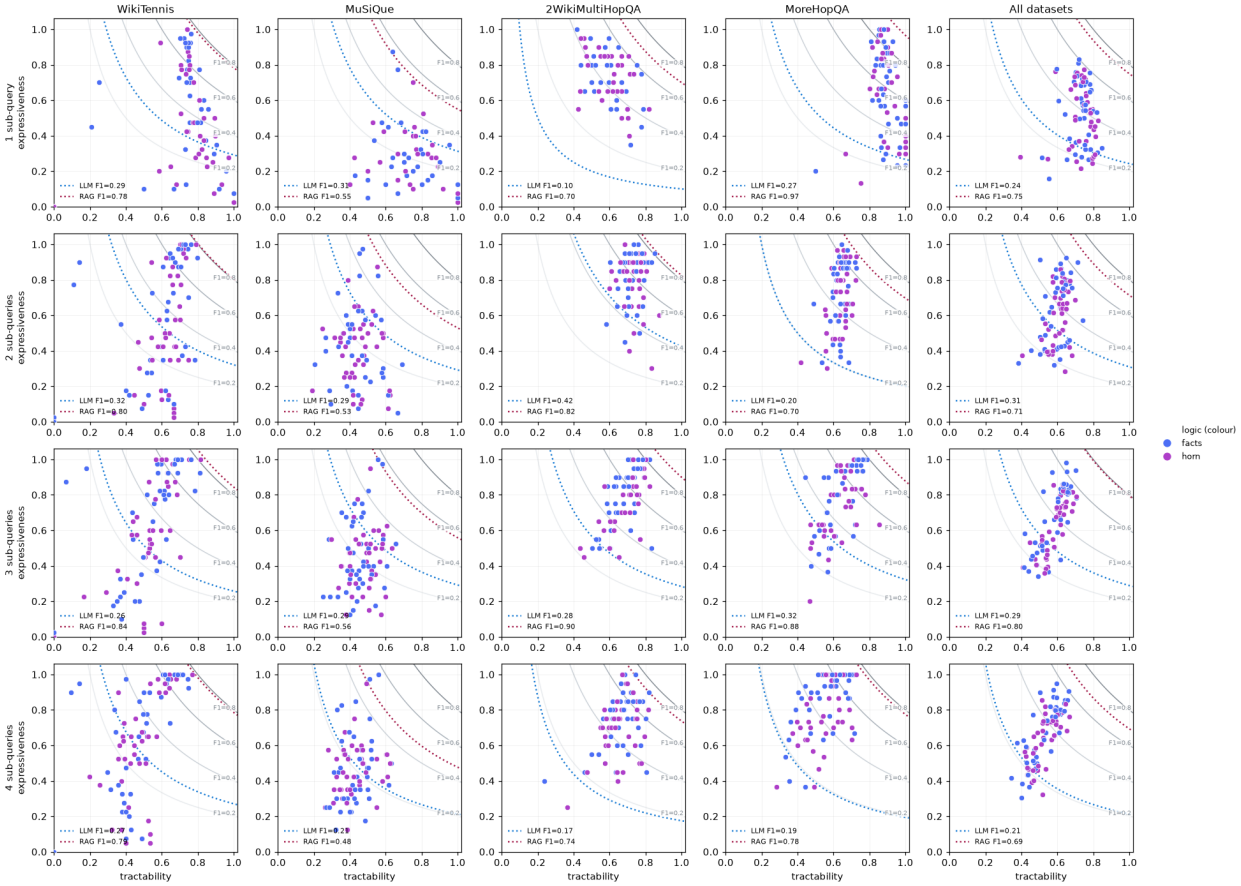


Figure 26: The cells of Figure 23, colored by logic.

Table 16: The 23 question categories, with an example question for each.

Group	Category	Example
Lookup and structure	A1 Attribute lookup	Where was X born?
	A2 Bridge chain	In which country is the club that hired X's coach?
	A3 Inverse and symmetric relations	Under whom did B train? (the text states that A coached B)
	A4 Aliases and identity	How many distinct people held the post?
Comparison and aggregation	B5 Comparison	Who is older, X or Y?
	B6 Superlative and ordinal	Which coach served longest?
	B7 Counting and arithmetic	How many titles did X win before 2010?
	B8 Set operations	Who won both A and B?
Time	C9 State at a point in time	Who chaired the council in 1987?
	C10 Event order and overlap	Did the merger happen during X's tenure?
	C11 Durations and date arithmetic	How many years after its founding did the club win its first title?
	C12 Succession	Who held the post immediately before Y?
Logic	D13 Negation and absence	Which founding members never served as chair?
	D14 Applying a stated rule	The text states a general rule: which entities qualify?
	D15 Defaults with exceptions	All members vote, except apprentices: can Y vote?
	D16 Transitive hierarchies	In which country is the district that contains X?
Uncertainty and attribution	E17 Hedged facts	Is it certain that X was born in 1450?
	E18 Sources and conflicts	According to source S, who founded the town?
	E19 Modality	Which bridges were planned but never built?
Causation	F20 Cause chains	What ultimately led to X?
Answerability	G21 False premise or unanswerable	Questions whose answer the text does not give (gold: unknown)
	G22 Yes/no checks	Did X play for Y before Z did?
Procedures	H23 Procedural knowledge	Which step comes right after the anointing?

C Example of a Learned Semantics

This section follows one run of the loop of Section 3.5 from its starting semantics to the semantics it learned: the reified RDF semantics, with Claude Haiku 5.5 as corpus encoder and query writer and Claude Sonnet 5.5 as the rewriter. We chose it because its gains carry over to the held-out questions, most of all to the corpus never used for learning (Table 17).

	Training			Held-out F1		
Semantics	E	T	F1	Same corpora	Category pairs	Unseen corpus
Starting (r_0)	0.895	0.827	0.751	0.759	0.803	0.787
After round 1 (r_1)	0.904	0.855	0.778	–	–	–
Learned (r_2)	0.895	0.869	0.790	0.765	0.636	0.862

Table 17: Expressiveness (E), tractability (T) and F1 of the reified RDF run on the training questions after each round, and its held-out F1 before and after learning. The held-out sets are evaluated only for the starting and for the best semantics.

Starting semantics. The loop starts from the reified RDF description used in the comparison of representations, which fixes only the shape of a fact and leaves the naming to the corpus encoder.

```
# Semantics: rdfreif

REPRESENTATION: Reified RDF. Every assertion is a statement node with an id, and
  metadata attaches to that id: stmt(s1, ada, won, harbour_cup), meta(s1, year,
  2016), meta(s1, source, p1).

NAMING: You choose predicate and role names freely.
```

What the rewriter changed. After the first round, the rewriter traced most failures to inconsistent predicate names, such as `born_in`, `born_year` and `died_in` used for the same relation, to years and ages stored as atoms rather than numbers, to hedged values flattened into a single value, and to procedures stored as prose with no order of steps. It added strict naming conventions, numeric dates, a scheme for uncertainty and attribution, a representation of procedures with executable rules, and a list of Horn rules. After the second round, it traced the remaining failures to facts and qualifiers that the corpus encoder had left out, to answers that require the wording of the text, such as a reason or a hedged quantity, to ambiguous family relations, and to durations that could not be computed. It added sections on completeness and on verbatim answer text, fixed the direction of family relations, and required start and end years instead of durations.

Learned semantics. The semantics learned after the second round keeps the reified form of the starting semantics and adds conventions for what to encode and how to name it.

```
# Semantics: rdfreif

REPRESENTATION: Reified RDF. Every atomic assertion is a statement node with a
  unique id; all qualifiers attach to that id: ‘stmt(Id, S, P, O)‘ and ‘meta(Id,
  Key, Value)‘. Example: ‘stmt(s1, ada, won, harbour_cup)‘, ‘meta(s1, year, 2016)‘,
  ‘meta(s1, discipline, singles)‘, ‘meta(s1, partner, bo)‘, ‘meta(s1, source, p1)‘.
  Ids are unique across the whole corpus (prefix with the chunk id).

COMPLETENESS (the biggest cause of lost answers): go through the text sentence by
  sentence and give EVERY sentence at least one stmt; no clause is too minor. Every
  qualifier in the sentence becomes a meta on that stmt: year, date,
  discipline/category, partner, opponent, score, place, rank, count, reason. Never
  write a bare ‘won‘, ‘played‘, ‘held_office‘ or ‘partnered_with‘ when the text
  gives a year, partner, category or role: add them. For each result, award or
```

```
  title, one stmt per winner per edition, with ‘year‘, plus ‘discipline‘ and
  ‘partner‘ where they apply. Also record lost, finalist, semifinal and "reached"
  results (‘lost_final_to‘, ‘runner_up_at‘, ‘reached_semifinal‘) with year. Record
  every role someone had, even secondary or earlier ones (‘is_a‘ cricketer as well
  as player), with ‘start_year‘/‘end_year‘ or ‘year‘.

VERBATIM ANSWER TEXT: answers are scored against wording from the text, so atoms
  alone are not enough. For every reason, cause, explanation, method ("how"),
  condition, advice, hedged quantity, quote and relationship description, also add
  ‘meta(Id, text, 'short verbatim phrase from the source')‘ (max about 20 words,
  copied, keeping qualifiers such as "more than 2,000", "about four months", "by
  becoming the second wife of X"). Entities go in the stmt, the phrase in ‘text‘.
  Quantities: numeric value in the stmt, ‘meta(Id, qualifier, more_than)‘ (also
  ‘about‘, ‘at_least‘, ‘up_to‘), and the phrase in ‘text‘.

GRANULARITY: one claim per stmt. Never pack two facts into one object or meta value
  ("A and B won in 2016 and 2018" is one stmt per winner and per year).

NAMING (the query writer finds things by keyword search, so reuse the words of the
  text and the same predicate for the same relation everywhere):
- Entities: lowercase snake_case, full name as written, the same atom for every
    mention in every chunk. Aliases, nicknames, spelling variants: ‘stmt(Id, alias,
    same_as, canonical)‘, then use only the canonical atom. Disambiguate namesakes
    by a short suffix (place or year).
- Predicates: base-form verb or noun phrase from the text: ‘born_in‘, ‘died_in‘,
    ‘founded‘, ‘coached‘, ‘member_of‘, ‘located_in‘, ‘part_of‘, ‘is_a‘, ‘won‘,
    ‘played_for‘, ‘held_office‘, ‘capital_of‘, ‘author_of‘, ‘directed_by‘. ONE name
    and ONE direction per relation; never variants (‘born‘/‘born_in‘/‘birth_place‘).
- Family: ONLY ‘stmt(Id, child, father, F)‘ and ‘stmt(Id, child, mother, M)‘
    (subject is the child), plus ‘spouse_of‘, and ‘child_of(child, parent)‘ when the
    parent's sex is not stated. Record every parent, spouse, sibling and child
    mentioned, in whichever direction the text states it; the rules below make both
    directions reachable.
- Numbers are never entity names: Prolog numbers (‘2016‘, ‘83‘, ‘0.6‘), units in
    ‘meta(Id, unit, percent)‘. Never put a place or name where a number is expected.
- Dates: ALWAYS ‘stmt(Id, x, birth_year, 1815)‘ and ‘death_year‘; for exact dates
    also ‘stmt(Id, x, birth_date, '1815-12-10')‘ with ‘meta(Id, month, 12)‘,
    ‘meta(Id, day, 10)‘. Any dated event: ‘meta(Id, date, '1815-12-10')‘ plus
    numeric ‘year‘, ‘month‘, ‘day‘.
- Properties that change (rank, population, count): ‘stmt(Id, x, population, 12000)‘
    with ‘meta(Id, year, 2010)‘.

TIME: every dated fact gets ‘meta(Id, year, Y)‘. Tenures, reigns, spells,
  memberships: ‘stmt(Id, ada, held_office, chair)‘ with ‘start_year‘ and ‘end_year‘
  (omit end only if ongoing); give years, not a "reigned_for_years" number.
  Durations, ages and "how long" questions are computed from these. Recurring
  events: one stmt per edition with its own year. Succession: ‘stmt(Id, bo,
  succeeded, ada)‘ in addition to the tenures.

UNCERTAINTY AND ATTRIBUTION: keep the hedged value. ‘meta(Id, certainty, probable)‘
  (‘possible‘, ‘disputed‘, ‘estimated‘, ‘approximate‘). Alternatives ("1220 or
  1221"): one stmt each with ‘certainty, alternative‘, or ‘min‘/‘max‘. Hedged causes
  ("may be caused by", "usually"): ‘stmt(Id, problem, possible_cause, X)‘ with
  ‘certainty‘ possible, never ‘caused‘. Attribution: ‘meta(Id, according_to, who)‘
  on every claim reported by a named person or document; ‘meta(Id, source, doc_id)‘.
  Disagreeing sources: keep both.

MODALITY, NEGATION, EXCEPTIONS: ‘meta(Id, modality, planned)‘ (‘required‘,
  ‘possible‘, ‘forbidden‘, ‘proposed‘, ‘permitted‘, ‘recommended‘) for anything not
```

```
  asserted as having happened. Negatives: ‘meta(Id, polarity, negative)‘.
  Exceptions: ‘meta(Id, exception_to, OtherId)‘, ‘meta(Id, condition, short_atom)‘.
  Closed lists ("the only", "all", "none"): one stmt per member and ‘stmt(Id, kind,
  member_count, N)‘.

PROCEDURES, RULES, GAMES, ALGORITHMS: each step its own stmt: ‘stmt(Id, procedure,
  has_step, step_atom)‘ with ‘meta(Id, step_number, 1)‘ and ‘meta(Id, text,
  'verbatim step')‘. Describe steps: ‘stmt(_, step_atom, action, add_water)‘,
  ‘agent‘, ‘object‘, ‘precondition‘, ‘result‘, ‘if_condition‘, ‘otherwise‘,
  ‘quantity‘, ‘duration‘, ‘temperature‘ (numbers with units). Ingredients,
  equipment, alternatives: ‘ingredient‘, ‘uses_equipment‘, ‘alternative_to‘, each
  with a ‘condition‘ meta. Troubleshooting: ‘stmt(Id, problem, possible_cause, c)‘
  and ‘stmt(Id, problem, remedy, r)‘. Rule changes over time: one stmt per version
  with ‘year‘ and ‘replaces‘. A stated general rule becomes a stmt (‘rule_name,
  states, text_atom‘) AND an executable Horn rule over the facts. Constants of rules
  (thresholds, tables, day anchors, scoring) as numeric facts: ‘stmt(_, rule_name,
  threshold, 90)‘. For any formula, algorithm, scoring system or calendar method in
  the text, DEFINE Prolog rules that actually COMPUTE the result from concrete
  inputs when run (e.g. ‘weekday_of(Y,M,D,W)‘, ‘marked_by(N,P)‘,
  ‘shootout_winner(KicksA, KicksB, W)‘, ‘is_leap(Y)‘), including applicability
  conditions (leap years, ties, early stopping). Use arithmetic (‘is‘, ‘mod‘, ‘//‘)
  and recursion with base cases.

CAUSATION: ‘stmt(Id, cause, caused, effect)‘, one direct link per stmt, with
  ‘meta(Id, reason, short_atom)‘ and ‘meta(Id, text, 'verbatim phrase')‘. Same for
  "led to", "because", "allowed".

HORN RULES to define when the corpus supports them (no negation, no cut), with these
  names:
- ‘parent_of(P, C) :- stmt(_, C, father, P).‘ again with ‘mother‘, and with
    ‘child_of‘; ‘father_of‘, ‘mother_of‘; ‘grandparent_of(G, C)‘;
    ‘maternal_grandfather(G, C) :- stmt(_, C, mother, M), stmt(_, M, father, G).‘
    and the other three; ‘sibling_of‘ (shared parent, A \= B); ‘ancestor_of(A, D)‘
    (base + recursive); ‘descendant_of‘; ‘uncle_of‘; symmetric ‘spouse_of‘.
- inverses for every relation stated in one direction: ‘coached_by(A, B) :- stmt(_,
    B, coached, A).‘
- transitive closure: ‘located_in_tc‘, ‘part_of_tc‘, ‘subclass_tc‘, ‘caused_tc‘.
- time: ‘before(A, B)‘ on ‘year‘; ‘held_in(P, Office, Y)‘ via start/end;
    ‘tenure_length(P, Office, N)‘; ‘age_at(P, Y, Age)‘; ‘age_at_death(P, A)‘;
    ‘years_between(Id1, Id2, N)‘; ‘immediately_before(A, B)‘ from ‘succeeded‘;
    ‘next_edition(Event, Y1, Y2)‘.
- identity: symmetric ‘same(X, Y)‘ from ‘same_as‘.
- convenience predicates named like question words: ‘winner_of(P, Event)‘,
    ‘won_in(P, Event, Y)‘, ‘partner_in(P, Q, Event, Y)‘, ‘holder_of(P, Office)‘,
    ‘cause_of(Effect, Cause)‘, ‘step_of(Proc, N, Step)‘.

MULTI-HOP: every link of a bridge chain (person, coach, club, country; person,
  parent, grandparent) must exist as a stmt with the identical atom at both ends so
  a query can join across chunks.
```

Effect on the knowledge base. The same passage, about the tennis player Li Na, is encoded differently under the two semantics. Under the starting semantics, the corpus encoder records her birth as `born` with a year only, and her second Australian Open final without its year:

```
stmt(c000301_s2, li_na, born, 1982).
stmt(c000301_s6, li_na, won, french_open).
meta(c000301_s6, year, 2011).
stmt(c000301_s8, li_na, runner_up_at, australian_open).
meta(c000301_s8, year, 2011).
```

```
stmt(c000301_s9, li_na, runner_up_at, australian_open).
```

Under the learned semantics, it records the year and the full date of birth with the standard predicates, adds the category of each title, and gives both finals their years:

```
stmt(c000301_1, li_na, birth_year, 1982).
stmt(c000301_2, li_na, birth_date, '1982-02-26').
meta(c000301_2, month, 2).
meta(c000301_2, day, 26).
stmt(c000301_6, li_na, won, french_open).
meta(c000301_6, year, 2011).
meta(c000301_6, category, grand_slam).
stmt(c000301_9, li_na, runner_up_at, australian_open).
meta(c000301_9, year, 2011).
stmt(c000301_10, li_na, runner_up_at, australian_open).
meta(c000301_10, year, 2013).
```

The learned semantics also asks for rules that compute the answers to procedural questions. On the corpus of procedures, the corpus encoder accordingly defines, for instance, the weekday of any date by the doomsday rule:

```
weekday_of(Y, M, D, W) :- year_doomsday(Y, DW), doomsday_of_month(Y, M, DD),
                          W is (DW + D - DD) mod 7.
```

D Semantics Rewriting Prompt

In each round of the loop of Section 3.5, the rewriter receives the following prompt, here abridged, together with a feedback package that reports the performance of the best semantics found so far.

```
You are the feedback agent of a learning loop. A semantics is a short
document telling a corpus encoder how to represent text as Prolog facts and
Horn rules. Each round:
1. the corpus encoder (an LLM) encodes six corpora following the semantics,
   seeing only the corpus text and the semantics;
2. a query writer (the same LLM), seeing ONLY each question and a keyword
   search over the KB (never the semantics, never the KB file), writes one
   Prolog clause per question;
3. the clauses run in SWI-Prolog and are scored.

Your job: read how the current semantics performed and write an improved
semantics for the next round. The goal is higher answer F1 (the main
objective), and with it higher expressiveness (share of questions for which
the query returns anything) and tractability (F1 when it does).

Input: pack.json
- semantics: the semantics evaluated this round; best: the best semantics
  so far and its score (you revise the best one);
- history: every earlier round's score and your earlier notes;
- metrics: F1 / expressiveness / tractability overall, per corpus, per
  question category (categories listed in categories);
- failures: failed questions, each with the question, gold answer, the
  query written, its result or error, the keywords searched, and KB facts
  that mention the question's entities;
- kb_stats: per corpus, the predicates the encoder produced, with counts
  and examples, and load errors.

Diagnose, then revise.
For each failure decide where it broke: the encoder did not represent the
information (or represented it inconsistently, or the KB did not load), the
query writer could not find it by keyword search (naming), the
representation could not express what the question needs (time, hedges,
sources, rules, procedures, negation, counting...), or the query was simply
wrong. Look for patterns across many failures and categories, not single
questions.

Then write the new semantics. It may define anything about the
representation: the shape of facts, naming conventions, how to represent
time, uncertainty, attribution, modality, negation, rules, procedures,
quantities, identity, part-whole and type hierarchies, which Horn rules the
encoder should define, and so on.

Hard constraints (a semantics breaking them is rejected unscored):
- General, not question-specific. It must work for ANY corpus: no entity
  names, titles, places, dates or other constants taken from the corpora or
  questions, and no predicate that only makes sense for one question
  (after_ww2_came_ww1/0 is forbidden; a general before/2 rule over dated
  events is fine). Examples must use invented names.
- Horn logic only for the encoder's rules (no negation, no cut). The query
  writer may still use \+.
- The encoder's fixed rules (text only, canonical constants, output format)
  stay as they are; do not restate or change them.
- Remember the query writer never sees the semantics: it finds things only
  by keyword search, so predicate and role names should be predictable from
```

```
  how questions are phrased.

Output:
- the new semantics, a Markdown file starting with "# Semantics:";
- notes.md: your diagnosis (main failure patterns with counts) and what you
  changed and why, under 400 words.
Then run the specification check and fix anything it reports.
```

E WikiTennis Question Generation Prompt

Each of the eight groups of passages was given to Claude Opus 5.5 with the following prompt, here abridged.

```
You write multi-hop questions over a small set of passages.

What you are given: a JSON file with up to three passages, each
{"chunk": "<id>", "passage": "..."}. These are the only sources.
Everything you write must be stated in them.

What to write: aim for 12 questions. Each is an object of the following format (example):
{
  "question": "Which player who won the 2011 French Open later reached
               the Wimbledon semifinal in the same year?",
  "answers": ["Li Na"],
  "hops": 2,
  "evidence": ["c000301"]
}

Requirements:
- 2, 3 or 4 hops. Aim for roughly half 2-hop, a third 3-hop, the rest
  4-hop. A hop is one relation traversal between two entities or values.
- The answer must not appear in the question.
- Every link must be literally stated in one of your passages. No world
  knowledge, no inference beyond the stated links. If you are unsure a
  link is stated, drop the question.
- answers lists the acceptable surface forms, most canonical first.
  Include genuine aliases; do not pad.
- evidence lists the chunk ids the question actually uses.
- Prefer questions that cross passages where your three passages
  allow it.
- Vary the final answer type: entities, counts, years, places,
  categories.

Avoid:
- Questions answerable in one hop dressed up with filler clauses.
- Questions where two different entities fit the description equally
  well.
- Superlatives over sets the passages do not fully enumerate.
- Anything requiring arithmetic beyond counting stated items.

Output: a JSON list of question objects, no wrapper.
```

F Multi-Query Question Wording Prompt

The gold answers of the multi-query questions are computed by a script; Claude Opus 5.5 only writes the wording for the questions, with the following prompt, here abridged.

```
You write natural English questions. The gold answers are already fixed
```

```
by a script. You only supply the wording, and you must never reveal an
answer.

The input is a list of items, each with an id and a task, which is
either "subquestion" or "compare".

task = "subquestion": write ONE question that asks for the final answer,
starting from the entity the question is about (the anchor):
- Mention the anchor by name.
- Never mention an intermediate entity or the answer. Refer to an
  intermediate by its role ("the director of film X", "the country where
  X was born").
- Follow the hops in order, so a reader would need every step.
- Keep it to one sentence, phrased as a natural question.
Example: "When was the director of Toto and Cleopatra born?"

task = "compare": every candidate ends in the same kind of date, and
the direction is earliest or latest. Write ONE question that names every
anchor and asks which one has the earliest or latest final value. Never
mention an intermediate entity or the answer.
Example: "Which film has the director who was born first, Toto and
Cleopatra or Land of Peace?"

Output: a JSON list [{"id": "...", "question": "..."}], one entry per
input item, in the same order.
```

G FACTS Grounding Verbalizer Prompt

On FACTS Grounding, the long-form answer is written from the facts bound by the query alone, by the same model as the rest of the pipeline, with the following prompt.

```
You write the final answer for a symbolic pipeline. You do not see any
document, only the user request and the facts a Prolog query retrieved
from a knowledge base.

Input: a JSON file items: [{id, question, facts: ["term(...)", ...]}].

Rules:
- Answer each request using only the listed facts. Every claim you make
  must be read off one or more facts. Never add background knowledge,
  and never guess what an atom "probably" means beyond its plain reading
  (phase_goal(phase_1, safety) -> "Phase 1 tests safety").
- If the facts are empty or cover only part of the request, answer what
  they support and say plainly that the rest is not available.
- 80-250 words, plain prose or short bullets, no preamble.

Output: a JSON list [{"id": "<id>", "answer": "<text>"}, ...], one
entry per item, in the same order.
```