



Representation Abstraction and Proof Search for English-to-Logic Question Answering

Tanel Tammet

Tallinn University of Technology

Abstract

We report experiments with *llmpipe*, our new English-to-logic question-answering pipeline. The pipeline translates English assertions and a question to a controlled first-order representation using both an LLM and a deterministic post-translation converter. The converter normalizes the LLM-produced formula, adds atoms and axioms, produces clauses, and calls the GK prover.

The experiments compare two evaluation settings. NLFORMTASKS is our 1600-case suite of short commonsense questions with minimal ambiguity. It uses tense, world state, exceptions, graded properties, modal distinctions and entity answers. FOLIO is a 203-case first-order deductive benchmark whose English sometimes needs lexical or background normalization, but usually not the fine-grained tense, world-state, defeasible-default or entity-answer constructions above. Our default representation is well-suited for NLFORMTASKS, but ill-suited for FOLIO. We develop and test representation variants with different abstraction granularity that either remove details from the generated clauses or add missing information.

Keywords. English-to-logic translation, automated theorem proving, knowledge representation, commonsense reasoning, proof search, large language models.

1 Introduction

We use *English-to-logic system* as our descriptive term for a system that translates natural language to a formal representation and then calls a prover [14, 13, 30, 3]. A growing family of such systems uses a large language model (LLM) as one part of the translation pipeline. Our paper describes a new English-to-logic pipeline *llmpipe* geared towards formalizing commonsense reasoning, along with the new 1600-case test set NLFORMTASKS we have built and used while developing it. The focus of the paper is a study of the deterministic conversion from an LLM-produced first-order representation to clauses which are given to the prover.

Our baseline translation uses an LLM twice: first to produce atomic semantic units, and then to translate these units to a controlled extended first-order representation. Here *controlled* means a fixed predicate inventory and *extended* means defeasible rules and numeric confidence annotations added to first-order logic. The deterministic part then adds type atoms, population witnesses, answer-definition predicates, context terms, measurement terms, definite-description

terms and selected axioms. It clausifies the result and calls GK, a saturation-based first-order prover extended with prioritized defaults, numeric confidences and contradiction-tolerant search [26, 24, 25].

Whether this detailed representation helps depends on the questions asked. It is useful for questions where the expected answer depends on fine-grained details like tense, world state, exceptions, event roles, modality or concrete entity answers. It is not always useful for abstract deductive benchmarks like FOLIO: in the latter, items are closer to textbook first-order reasoning, with proof often using class information and simple relations, not detailed event roles, contexts, world states and defeasible reasoning machinery. For such items our standard conversion builds a clause set that is sometimes too detailed and sometimes missing the abstractions FOLIO benefits from.

We develop representation conversion mechanisms for both abstracting from the fine-grained representation towards a more coarse-grained representation and for generating shorter proofs. During the experiments described in the paper the prover and base search settings are fixed; the post-translation variants may change the generated atoms and the additional axioms selected for a case. We report two groups of experiments.

1. Comparing the full representation and several post-conversion variants on NLFORMTASKS (NLFT in the rest of the paper) and FOLIO. Results are given separately for GPT, Claude, Gemini and DeepSeek.
2. Measuring proof length separately from answer accuracy. We compare a lossy event simplification, a compact Davidsonian [4] event form, and a narrow existential-attribute collapse.

The paper is an experimental and tool paper. Data, prompts, per-case generated clauses, prover inputs/outputs, and grading scripts: the `lpar` branch of <https://github.com/tammet/nlformtasks> (checkout: `git checkout lpar`). The pipeline is <https://github.com/tammet/nlpsolver> at tag `lpar-2026-06-23`.

2 Related Work

llmpipe belongs to what we call the *LLM-to-solver* line: an LLM formalizes the problem and a symbolic engine performs the inference. The line is represented by Logic-LM [14] and its refinement Logic-LM++ [7], LINC [13], SatLM [30], CLOVER [19], VERUS-LM [3], LINA [10], MALLS [29] and PEIRCE [17]. Those systems differ in target logic and solver. Logic-LM uses solver errors to repair the formalization, Logic-LM++ adds backtracking, and LINC votes over several sampled formalizations. Two recent papers vary the solver side: Xu et al. [28] choose and compose the logical solver per problem, and Feng et al. [5] let an LLM propose auxiliary lemmas for constraints with inductive definitions, which a solver then validates. The focus of our paper is narrower: after the English has been translated to a fixed controlled representation, how should that representation be converted to clauses?

Deductive natural-language reasoning is commonly evaluated on FOLIO [6], with human-written English and first-order annotations, and on the synthetic ProofWriter [23] and PrOntoQA [20]. Our NLFT suite instead targets tense, world states, exceptions, confidences and entity-valued answers.

The intermediate representation is known to matter [2]. A naive LLM-to-FOL-to-prover path can also underperform direct prompting when the formulas are brittle or incomplete [12]. Our FOLIO results are consistent with this: the standard commonsense representation is not the best conversion for a FOLIO-style proof.

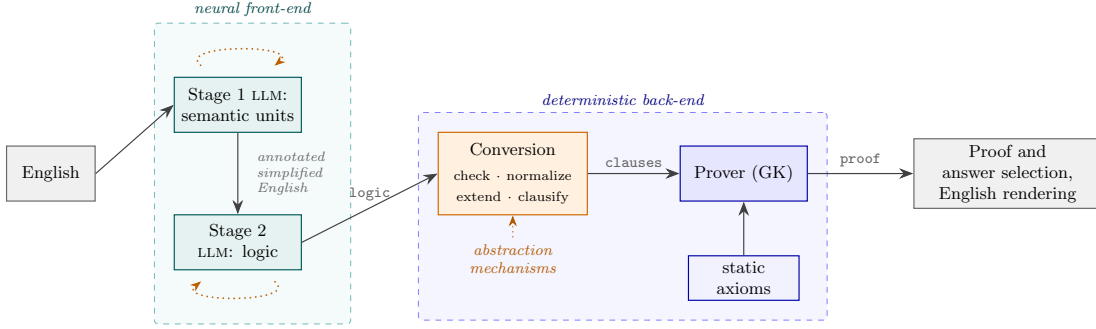


Figure 1: The *llmpipe* pipeline. A language model is called twice — Stage 1 maps English to annotated semantic units, Stage 2 maps those to logic over a controlled predicate inventory — and everything after Stage 2 is deterministic code. Each LLM stage is followed by a structural sanity check that can re-call the model (dotted loops). The deterministic converter applies the post-translation variants studied in this paper (Table 2) before clausifying and handing the clause set, together with a static axiom file, to the GK prover.

The default *llmpipe* representation uses classical devices. Event reification follows the Davidsonian [4] and neo-Davidsonian [15] traditions; defaults follow Reiter-style default reasoning [18]; world and frame reasoning are related to the situation and event calculus traditions [11, 8]. Episodic Logic [22], Cyc [9], probabilistic worldbuilding [21] and ontology-grounded English-to-logic work such as HyNOLU [16] are broader relatives.

We note that defeasible natural language reasoning can also be implemented in ASP or s(CASP) [1]. We use an extended first-order prover rather than an ASP solver. Some of the representation failures we report, such as losing event roles or over-generating type atoms, are solver-independent.

3 Pipeline and Post-translation Conversion

Only the first two stages of the baseline translation call an LLM. The prompts for both stages consist of about 50 KB of instructions and about 50 KB of examples. Everything after Stage 2 is deterministic code. We use the non-reasoning configuration of each model: GPT-5.1 with reasoning effort set to *none*, Claude Sonnet 4.6 with extended thinking disabled, and DeepSeek non-reasoning chat model v4-flash; Gemini 2.5 Flash is an exception, as it retains the dynamic “built-in” thinking that its API applies by default. LLMs are called at temperature 0 (not sent on GPT’s API path), with responses cached so runs are reproducible for a pinned model set. The default GK prover time limit is 2 seconds per question in all configurations; however, for a few multi-world clause sets in NLFT the time limit is heuristically increased.

Stage 1. Each semantic unit (ASU) has a type: timeless fact, situation, strict rule, defeasible rule or query. The annotations contain entity records, adjective records, optional confidence and action records. Entity records distinguish concrete and generic entities and may include a broad category such as **person**, **animal**, **artifact** or **place**. Action records contain a verb root, roles such as actor, target, recipient and location, and a mode such as event, habitual, capability, obligation, volition or speech act. The roles are semantic roles, not grammatical

cases: the actor of a passive sentence is still its actor. Situation-changing actions also carry pre- and post-world identifiers.

Stage 2. The second LLM call maps the ASUs to a controlled and extended first-order representation. The predicate inventory is fixed. The main predicates are `isa`, `has property`, `is rel2`, `have`, `has part`, degree variants for gradable properties and relations, and event-role predicates. Dynamic verbs are represented as *reified neo-Davidsonian* events: the event variable is the central object and every participant is attached by a separate thematic-role literal, instead of being an argument of the verb predicate as in the original Davidsonian form. For example

$$\text{isa}(\text{activity}, E) \wedge \text{has_type}(E, \text{fly}) \wedge \text{has_actor}(E, X) \wedge \text{capability}(E),$$

where the event `E` carries its verb as a value (`has_type`), its participants as role literals (`has_actor`), and an optional modal classifier (`capability`). Actual events create new worlds. Rules and facts are wrapped by a world state indicator. Defeasible rules use a `normally` marker wrapping the internal first-order logic (FOL). Events may have modalities (capability, typical, intention, etc.). Questions are represented either as yes/no questions or as wh-questions with an answer variable.

Conversion. The Stage-2 formula is not sent directly to the prover: it is a representation containing wrappers, metadata-sensitive constructs and markers that the converter expands before clausification. The converter clausifies and adds material from both the formula and the Stage-1 metadata:

- entity categories and base words become `isa` atoms when needed;
- compounds, names, plurals and gendered nouns may add type or subsumption atoms in abstraction variants;
- universal rules may generate positive and negative population witnesses;
- sets, counting and measures along with arithmetic functions may be introduced;
- questions are converted to internal answer definitions using a special `$defq` predicate;
- eligible atoms receive a context term `$ctxt(T,W,L,K)` for tense, world, location and knower;
- definite descriptions and measurements are rewritten to canonical function terms;
- the vocabulary and question form trigger an injection of different bridge, exclusion, frame, movement and transfer axioms;
- `normally` markers are expanded to blocker literals for prioritized default reasoning.

The GK prover and its input. GK extends the saturation-based prover GKC [26]. Its input is a clause list in the JSON-LD-LOGIC format [27]: a clause is a JSON object whose `@logic` value is a literal or a disjunction of literals; each literal is a list headed by its predicate; `-` marks negation; `?:X` is a variable and `#:Mike` a constant; the clause with `@question` is the goal. Before a default conclusion is accepted, GK searches recursively for a higher-priority counterargument to its blocker [24]; confidences are combined along derivations [25]; only proofs using a clause from the question are accepted, which tolerates contradictions elsewhere.

As an example, we trace “*Birds fly. Penguins are flightless birds. Mike is a bird and John is a penguin. Who can fly?*” with Claude (answer: *Mike*). Stage 1 makes “*Birds fly*” a `normal_rule` unit (text “*Birds can fly*”, a generic `animal` entity, root `fly`, mode `capability`, confidence 0.95) and splits “*Penguins are flightless birds*” into a strict subclass rule and a strict rule denying

the capability. Stage 2 uses the four event literals above to state that every bird X normally has a flying event E in capability mode. The rule is associated with world $W0$ and confidence 0.95; the question asks which X satisfy the same event conditions. The converter Skolemizes the event to $sk0(X)$, distributes the rule into one clause per conjunct, expands **normally** to a blocker literal and maps the confidence p to the evidence $2p - 1$. The first and the last of the four rule clauses and the question reach GK as (context terms collapsed to $\$c$):

```
{ "@name": "sent_S1", "@nl": "Birds can fly.",
  "@logic": [[ "-isa", "bird", "?X"], [ "isa", "activity", ["sk0", "?X"] ] ] },
{ "@name": "sent_S1", "@confidence": 0.9,
  "@logic": [[ "-isa", "bird", "?X"], [ "capability", ["sk0", "?X"] ],
    [ "$block", ["$", "bird", 1], ["$not", ["capability", ["sk0", "?X"] ] ] ] ] },
{ "@name": "sent_S6", "@question": ["$defq0", "?X"], "@askvars": 1 }
```

Further clauses give the event’s type and actor, define $\$defq0(X)$ and encode the penguin rules. GK obtains Mike as an answer in a proof with 9 derived clauses under the blocker and fails to derive the exception “*the flying event of Mike is not a capability*”, so the answer stands.

4 Evaluation Settings

NLFT contains 1600 short English question-answering cases with minimal ambiguity. Each case consists of assertions and one question. Answers may be *True*, *False*, *Unknown*, a confidence-qualified answer such as *probably true*, or a (possibly also confidence-qualified) concrete answer such as an entity, place, time or value. A few examples:

Birds fly and eat. Baby birds do not fly. John is probably a baby bird. Mike and Eve and John are birds. Who flies and eats? Expected: Mike and Eve. John stopped smoking. Did John smoke in the past? Expected: True. John stopped smoking. Does John smoke now? Expected: False.

The suite was built incrementally during the development of the pipeline. Most cases were written manually by the author; the rest were selected and adapted from earlier test sets and linguistic examples. The author proposed the expected answers and checked discrepancies with pipeline output by hand; cases where independent LLM answer proposals clearly diverged were excluded. A design goal was that under the default pipeline no case is failed by more than one of the four LLM backends; this holds in the published baseline run. The suite is therefore near-saturated by construction.

FOLIO has 203 validation items in the FOLIO v2 version used here [6]. It asks for *True*, *False* or *Unknown*. The language is often less direct than in NLFT. The intended reasoning is usually abstract first-order deduction. Most FOLIO items do not require tense, world states, defeasible defaults or entity-valued answers. However, they may require nontrivial world knowledge. Consider the following FOLIO example.

Premises: ***Symphony No. 9 is a music piece. Composers write music pieces. Beethoven wrote Symphony No. 9. Vienna Music Society premiered Symphony No. 9. Vienna Music Society is an orchestra. Beethoven leads the Vienna Music Society. Orchestras are led by conductors.*** **Conclusion:** ***Beethoven is a composer.***

The sentences relevant to the proof are in bold; the rest is ballast. The expected answer is *True*: FOLIO’s expected value requires treating *writing a music piece* as sufficient evidence for being a composer — a converse reading that our default pipeline does not add. Such a converse

Table 1: Baseline accuracy (%) without additional mechanisms.

Setting	GPT	Claude	Gemini	DeepSeek
NLFT	97	99	98	99
FOLIO	54	57	51	47

is not generally valid: compare *Composers write letters to managers*, where it would be wrong. When the four evaluation models are asked directly for the commonsense answer to the FOLIO questions, with no logic and no prover, their alignment with the expected FOLIO labels is GPT 69%, Claude 67%, Gemini 86%, DeepSeek 86% (first row of Table 3); Fable-5, asked the same way, reaches 91% over the 195 cases it answered (8 content refusals excluded).

The two suites therefore stress different aspects of the autoformalization problem. NLFT tests whether the representation preserves fine-grained semantic distinctions. FOLIO tests whether the representation is abstract enough.

We report *accuracy*: the percentage of cases whose answer the grader accepts. An answer is accepted if it matches the case’s expected value — or any of its listed acceptable alternatives — under a normalizing comparison that ignores case, punctuation and word order and tolerates preposition variants, unit conversions and, by default, confidence qualifiers (so “*Probably true.*” matches *True*, and “*In the office.*” matches *office*); a *None* expectation accepts any *Unknown* answer. Tables keep the four LLM backends separate; where the text names one backend, we use Claude as the default example.

5 Adjusting the Representation Granularity

Table 1 shows that on NLFT, our pipeline is near saturation. On FOLIO, the same representation is much weaker. A significant number of pipeline errors with respect to the expected FOLIO answers are due to representation/conversion mismatches.

First, our representation keeps event structure and context, including time and world-state setting, even when a FOLIO proof only needs a simple predicate or class relation. The additional arguments and literals can lengthen clauses or block a match that would be available in a coarser form. Second, FOLIO often benefits from type completion: if the text uses a name, plural, compound noun or surface variant, the proof may need an added class atom or subsumption relation. The full NLFT setting is conservative about adding such atoms because over-generating them can create wrong definite answers. Third, FOLIO rewards abstraction over wording. In NLFT, the wording is intentionally less ambiguous and often tests a particular semantic distinction. In FOLIO, several correct proofs depend on treating surface variants as the same class or relation, as in the following example, which is answered “Unknown” by our pre-normalized default pipeline and “True” with the abstraction machinery:

Premises: *A Japanese game company **created** the game the Legend of Zelda. All games on the Top 10 list are **made by** Japanese game companies. If a game **sells more** than one million copies, then it will be included **in the Top 10 list**. The Legend of Zelda **sold more** than one million copies.* Conclusion: *The Legend of Zelda is **on the Top 10 list**.*

The recorded Claude translation shows three mismatches: the rule uses the relation **included in** but the question **on**; the Legend of Zelda is typed **artifact** while the rule is guarded by

Table 2: Post-translation variants.

Variant	Meaning
Pre-normalized base	FOLIO-only wording normalization by an extra LLM call before the usual two-stage translation.
Lossy event simplification	Replace a detailed reified event by a flat relation; secondary roles and adjuncts may be lost.
Extra type/class atoms	Add class and subsumption atoms from names, compounds, gendered nouns and supertypes; also simplify contexts and remove default blockers.
Entity merging	Merge some names, sets or surface variants judged likely to refer to the same object.
Relational definites	Keep definite descriptions as plain relations instead of rewriting them to canonical function terms.
Guard removal	After event simplification, remove type guards from rule bodies.
Extra bridges	Add bridge axioms between event, relation, frame and occasion forms.
FOLIO abstraction	A bundle of the FOLIO-oriented changes above, plus a conditional cross-stage re-translation.

`isa(game,X)`; and the condition *more than one million* is kept in the rule but dropped from the fact; each alone blocks the proof. Under the full FOLIO abstraction the pipeline re-translates the input once (the cross-stage retry) and obtains the game type, the count in both places and one relation name; the flat event fold drops the count and guard removal drops the game guard, reducing the rule body and the fact to `is_rel2(sell,X,copy)`.

These observations motivate the following switchable transformation mechanisms we have implemented on top of the default pipeline (Table 2).

The mechanisms work as follows. *Pre-normalization* is an additional LLM call that rewrites the English input into plainer wording before Stage 1. *Lossy event simplification* is the flat relational event fold of Section 6: an event with an actor and one object role becomes a binary `is_rel2` relation of the verb, the actor and the object, an event with an actor and no object becomes a unary property; the event variable, modal classifiers, adjuncts and further roles are dropped.

Extra type/class atoms adds `isa` atoms and subsumption axioms that the default conversion withholds: broad supertypes (person, animal) even when a subtype is present, a gender class from a first name or a gendered role noun, the name of a multiword proper name as its own type, and subsumption from every suffix of a compound noun. *Entity merging* unifies proper-name constants sharing a type and most of their words. *Relational definites* disables the default rewriting of definite descriptions into function terms.

Guard removal drops an `isa(T,V)` antecedent when `V` already occurs in a folded relation of the same antecedent or `T` is a near-universal type such as *thing*, which strengthens the rule. *Extra bridges* adds axioms linking a folded relation and a reified event of the same verb, and a few similar pairs. The FOLIO *abstraction* bundle applies all of the above, collapses degree literals, removes the default blockers, replaces structured contexts with one shared variable per clause and, at most once per case, re-runs Stages 1 and 2 with a corrective hint when a rule antecedent has no support in the clause set (the *cross-stage retry*). The bundle is therefore not

Table 3: FOLIO accuracy (%) for representation variants, over all 203 items. The first row is the direct LLM answer, with no logic and no prover. Parentheses give items *gained* (+) and *lost* (−) against the pre-normalized base (the reference row) on the same items; their difference gives the change in the number of correct answers. The highest pipeline accuracy in each column is in bold.

Variant	GPT	Claude	Gemini	DeepSeek
Direct LLM answer (no logic, no prover)	69	67	86	86
Pre-normalized base	56	57	52	47
Lossy event simplification	57 (+7/−5)	61 (+10/−3)	55 (+11/−5)	48 (+6/−5)
Extra type/class atoms	65 (+27/−9)	61 (+23/−15)	58 (+29/−17)	51 (+15/−8)
Entity merging	56 (+0/−0)	56 (+0/−3)	51 (+0/−1)	48 (+1/−0)
Relational definites	56 (+0/−0)	57 (+0/−0)	51 (+0/−2)	48 (+3/−1)
Lossy event simplification + guard removal	58 (+8/−5)	64 (+17/−3)	58 (+18/−5)	50 (+11/−6)
Lossy event simplification + extra bridges	57 (+7/−5)	60 (+10/−4)	55 (+11/−5)	48 (+7/−5)
FOLIO abstraction	65 (+25/−7)	70 (+35/−8)	67 (+41/−10)	53 (+20/−8)

the sum of the isolated mechanisms applied to one fixed translation.

The “+ guard removal” and “+ extra bridges” rows apply the flat fold and then one further change. For every row in the next two tables, gains and losses are counted against the table’s reference row, not against the flat fold alone.

Table 3 shows FOLIO. Its reference row, the *pre-normalized base*, includes the FOLIO-only pre-normalization step, so it is not identical to the FOLIO row of Table 1. The best row varies by backend, but the pattern is stable: abstraction helps. For Claude, the score goes from 57.1% in the pre-normalized base to 70.4% in the full FOLIO abstraction.

Among the mechanisms applied on their own, extra type/class atoms gives the largest net gain for each backend. Since this setting also simplifies contexts and removes default blockers, its gains cannot be attributed to type completion alone. Its net gain equals the bundle’s for GPT (18 items, with different items gained and lost) but is well below it for Claude, Gemini and DeepSeek (8, 12 and 7 against 27, 31 and 12).

We note that the best percentage-wise result for FOLIO (78%) is obtained from the “abstraction” results above by majority voting of all four LLMs, where the *Unknown* value abstains. This is similar to LINC [13].

The first row of Table 3 is the direct-answer reference of Section 4. The full FOLIO abstraction exceeds direct answering only for Claude (by six items) and stays below it for GPT, Gemini and DeepSeek; on FOLIO the pipeline does not consistently improve accuracy over direct answering. On the earlier 204-item FOLIO release, Logic-LM reports 78.9% with GPT-4 [14] and Logic-LM++ 84.8% [7], with other models, prompts and items.

Table 4 shows the same kind of variants on NLFT. Here the full representation remains best. The variants that help FOLIO either lose information or over-generalize. Entity merging and relational definites are nearly neutral.

Disabling the implementation’s `plural` sub-option raises Claude from 91% to 98% on the NLFT type/class row, but also disables context simplification and blocker removal; this comparison therefore does not isolate plural normalization.

Table 4: NLFT accuracy (%) for representation variants, over all 1600 items. Parentheses give items *gained* (+) and *lost* (−) against the full representation (the reference row) on the same items; their difference gives the change in the number of correct answers. The highest accuracy in each column is in bold.

Variant	GPT	Claude	Gemini	DeepSeek
Full representation	97	99	98	99
Lossy event simplification	91 (+3/−100)	93 (+0/−98)	92 (+3/−105)	91 (+0/−121)
Extra type/class atoms	89 (+9/−134)	91 (+6/−128)	90 (+8/−129)	91 (+2/−127)
Entity merging	97 (+1/−1)	99 (+0/−0)	98 (+0/−4)	98 (+0/−8)
Relational definites	97 (+1/−2)	99 (+0/−1)	98 (+3/−2)	99 (+1/−2)
Lossy event simplification + guard removal	91 (+3/−101)	93 (+0/−99)	92 (+3/−105)	91 (+0/−122)
Lossy event simplification + extra bridges	91 (+4/−104)	91 (+0/−120)	91 (+3/−118)	90 (+0/−133)

6 Proof Length: Event and Attribute Folds

For almost all NLFT and FOLIO examples that have proofs, our GK prover finds them fast. We use proof length as a proxy for proof-search cost. Throughout this section, *proof length* means the number of derived clauses in a proof, not counting the input clauses.

The full reified neo-Davidsonian representation can generate long proofs because one English verb becomes an event object plus several role literals. In classical notation, with world, tense and context arguments omitted, “John quickly drove the car” becomes

$$\exists e (activity(e) \wedge type(e, drive) \wedge actor(e, john) \wedge object(e, car) \wedge manner(e, quick)).$$

We have implemented and tested three changes to proof shape, shown below: the two event folds on this example, and a narrower existential-attribute collapse on a separate “has-a” pattern.

Flat event form. A lossy form that drops the event variable entirely and turns the event into a simple relation: *is_rel2(drive, john, car)*. It is short, but the event variable and the *quick* adjunct are gone.

Compact Davidsonian form. A Davidsonian atom $\exists e (event(drive, john, car, e) \wedge quick(e))$ that keeps the event handle but packs the verb, agent and patient into one positional literal, leaving every other role as an adjunct on *e*. The patient slot (here *car*) is the event’s theme: the first role present among *has target*, *has goal*, *has topic*.

Existential-attribute collapse. A narrow rewrite of a bare existential-attribute pattern, for example “*x* has a leg”, $\exists y (leg(y) \wedge part(x, y))$, into a structured property *hasLeg(x)*, tied to it by a biconditional bridge:

$$\begin{aligned} \forall x \forall y (leg(y) \wedge part(x, y) \rightarrow hasLeg(x)), \\ \forall x (hasLeg(x) \rightarrow leg(w(x)) \wedge part(x, w(x))). \end{aligned}$$

The first direction lets any leg-witness feed the property; the second recovers a single canonical witness *w(x)* — a dedicated function shared by all consumers rather than a fresh existential.

With Claude the flat form raises FOLIO from 57.1% to 60.6% on the pre-normalized base, but lowers NLFT from 98.9% to 92.8%. Its failures are easy to explain: a question may depend on a

Table 5: Claude only. For each case the baseline proved (proved, not necessarily answered correctly), we report how many of those proofs the variant makes *shorter*, *longer* or leaves the *same* length, and how often it found no proof (*lost*) or found one where the baseline had none (*gained*).

Suite	Variant	shorter	longer	same	lost	gained
NLFT	compact Davidsonian	416	43	695	12	5
NLFT	existential collapse	37	55	1067	7	0
NLFT	both	453	104	591	18	5
FOLIO	compact Davidsonian	26	4	31	2	2
FOLIO	existential collapse	5	0	56	2	2
FOLIO	both	29	4	27	3	4

Table 6: Proof length (Claude) on four long cases. *+Davidsonian* is the compact Davidsonian form, *+existential* the existential-attribute collapse, and *+both* the two combined. The NLFT cases use the standard encoding; the FOLIO case uses the max-abstract baseline.

Case	input excerpt	baseline	+Davidsonian	+existential	+both
NLFT 864	“John drove a car which Mike did not buy...”	28	5	28	5
NLFT 618	“Either John or Bill went to the store...”	47	18	47	18
NLFT 785	“Bears who are nice and white eat fish...”	28	18	28	18
FOLIO 198	“...if an animal has legs, then it jumps...”	164	92	14	9

role or adjunct that the flat form removes. The compact Davidsonian form is the safer general event shortener: with it, Claude scores 97.9% on NLFT and 57.6% on FOLIO. Table 5 reports, for Claude, how many baseline proofs become shorter or longer under each fold, counted only over cases where the baseline found a proof.

We illustrate the difference on two of the concrete cases in Table 6. NLFT 864, “*John drove a car which Mike did not buy. John drove a car Mike did not buy?*” (answer *True*), has two events, *drove* and *buy*. The full neo-Davidsonian proof derives 28 clauses; the compact Davidsonian form collapses the role literals onto the two verbs and proves it in 5. The existential collapse matches nothing here and leaves the proof unchanged.

FOLIO 198 is the complementary extreme case: “*Jane does not like any animals that bark. ... If an animal has legs, then it jumps. If an animal is terricolous, then it has legs. ...*” (answer *True*). The baseline proof has 164 clauses. Collapsing the attribute removes the witnesses and cuts the proof to 14; adding the compact Davidsonian form brings it to 9.

7 Summary and Ongoing Work

We have introduced a new experimental LLM-based English-to-logic autoformalization pipeline *llmpipe* with several novel capabilities. We have also introduced a 1600-case natural language question answering test suite NLFORMTASKS (NLFT in the paper) covering a number of linguistic and commonsense reasoning constructions not well covered in existing test suites. The experimental results presented in the paper compare the performance of *llmpipe* on the NLFT suite with its performance on the FOLIO suite. Various abstraction mechanisms as well as proof-shortening transformations to the representation are introduced and compared.

We continue experimenting with new and improved abstraction mechanisms. One promising approach is to keep the full representation and to retry only when the prover returns *Unknown*: first two deterministic re-conversions of the same parse, then an LLM critique with re-translation, then a second translation into open relation triples. In August 2026 runs on the same 203 FOLIO items this gives GPT 69%, Claude 75%, Gemini 69% and DeepSeek 71%, above the FOLIO abstraction row of Table 3 for every backend, with NLFT at about 98% for Gemini and 99% for DeepSeek. As for the compact Davidsonian form, a reversible variant that folds an event only when the fold can be undone exactly, together with a narrower existential collapse, caused no correct answer to become incorrect over all NLFT and FOLIO items and all four backends, while the event fold still shortened 189–250 shared correct NLFT proofs per backend.

References

- [1] Joaquín Arias, Manuel Carro, Elmer Salazar, Kyle Marple, and Gopal Gupta. Constraint answer set programming without grounding (s(CASP)). *Theory and Practice of Logic Programming*, 18(3–4):337–354, 2018.
- [2] Alexander Beiser, David Penz, and Nysret Musliu. Making LLMs reason? the intermediate language problem in neurosymbolic approaches. *arXiv:2502.17216*, 2025.
- [3] Benjamin Callewaert, Simon Vandeveld, and Joost Vennekens. VERUS-LM: A versatile framework for combining LLMs with symbolic reasoning. *arXiv:2501.14540*, 2025.
- [4] Donald Davidson. The logical form of action sentences. In *The Logic of Decision and Action*. University of Pittsburgh Press, 1967.
- [5] Weizhi Feng, Shidong Shen, Jiaxiang Liu, Taolue Chen, Fu Song, and Zhilin Wu. Can LLM aid in solving constraints with inductive definitions? In *Formal Methods (FM 2026)*, volume 16557 of *Lecture Notes in Computer Science*, pages 111–132. Springer, 2026.
- [6] Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Wenfei Zhou, James Coady, David Peng, Yujie Qiao, Luke Benson, Lucy Sun, Alexander Wardle-Solano, Hannah Szabó, Ekaterina Zubova, Matthew Burtell, Jonathan Fan, Yixin Liu, Brian Wong, Malcolm Sailor, Ansong Ni, Linyong Nan, Jungo Kasai, Tao Yu, Rui Zhang, Alexander Fabbri, Wojciech Maciej Kryscinski, Semih Yavuz, Ye Liu, Xi Victoria Lin, Shafiq Joty, Yingbo Zhou, Caiming Xiong, Rex Ying, Arman Cohan, and Dragomir Radev. FOLIO: Natural language reasoning with first-order logic. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 22017–22031, Miami, Florida, USA, November 2024. Association for Computational Linguistics.
- [7] Shashank Kirtania, Priyanshu Gupta, and Arjun Radhakrishna. LOGIC-LM++: Multi-step refinement for symbolic formulations. In Bhavana Dalvi Mishra, Greg Durrett, Peter Jansen, Ben Lipkin, Danilo Neves Ribeiro, Lionel Wong, Xi Ye, and Wenting Zhao, editors, *Proceedings of the 2nd Workshop on Natural Language Reasoning and Structured Explanations (@ACL 2024)*, pages 56–63, Bangkok, Thailand, August 2024. Association for Computational Linguistics.
- [8] Robert Kowalski and Marek Sergot. A logic-based calculus of events. *New Generation Computing*, 4(1):67–95, 1986.
- [9] Douglas B. Lenat and R. V. Guha. *Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project*. Addison-Wesley, 1990.
- [10] Qingchuan Li, Jiatong Li, Tongxuan Liu, Yuting Zeng, Mingyue Cheng, Weizhe Huang, and Qi Liu. Leveraging large language models for hypothetical deduction in logical inference: A neuro-symbolic approach (LINA). *arXiv:2410.21779*, 2024.
- [11] John McCarthy and Patrick J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence*, 4:463–502, 1969.

- [12] Bill Noble, Rasmus Blanck, and Gijs Wijnholds. In the mood for inference: Logic-based natural language inference with large language models. In *NALOMA Workshop*, 2025.
- [13] 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 Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5153–5176, Singapore, December 2023. Association for Computational Linguistics.
- [14] Liangming Pan, Alon Albalak, Xinyi Wang, and William Wang. Logic-LM: Empowering large language models with symbolic solvers for faithful logical reasoning. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3806–3824, Singapore, December 2023. Association for Computational Linguistics.
- [15] Terence Parsons. *Events in the Semantics of English: A Study in Subatomic Semantics*. MIT Press, 1990.
- [16] Adam Pease and Josef Urban. Towards open domain english to logic translation. In *AITP*, 2023.
- [17] Xin Quan, Marco Valentino, Danilo S. Carvalho, Dhairya Dalal, and André Freitas. PEIRCE: Unifying material and formal reasoning via LLM-driven neuro-symbolic refinement. In *Proceedings of ACL 2025: System Demonstrations*, 2025.
- [18] Raymond Reiter. A logic for default reasoning. *Artificial Intelligence*, 13(1–2):81–132, 1980.
- [19] Hyun Ryu et al. Divide and translate: Compositional first-order logic translation and verification for complex logical reasoning. In *ICLR*, 2025.
- [20] Abulhair Saparov and He He. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought (ProntoQA). In *ICLR*, 2023.
- [21] Abulhair Saparov and Tom M. Mitchell. Towards general natural language understanding with probabilistic worldbuilding (PWM). In *TACL*, 2022.
- [22] Lenhart K. Schubert. NLog-like inference and commonsense reasoning (episodic logic / EPILOG). *Linguistic Issues in Language Technology*, 9, 2014.
- [23] Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. ProofWriter: Generating implications, proofs, and abductive statements over natural language. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3621–3634, Online, August 2021. Association for Computational Linguistics.
- [24] T. Tammet, D. Draheim, and P. Järvi. GK: Implementing full first order default logic for commonsense reasoning (system description). In J. Blanchette, L. Kovács, and D. Pattinson, editors, *IJCAR 2022: Automated Reasoning*, volume 13385 of *LNCS*, pages 300–309. Springer, 2022.
- [25] T. Tammet, P. Järvi, and D. Draheim. Confidences for commonsense reasoning. In A. Platzer and G. Sutcliffe, editors, *Automated Deduction – CADE 28*, volume 12699 of *LNCS*, pages 507–524. Springer, 2021.
- [26] Tanel Tammet. GKC: A reasoning system for large knowledge bases. In Pascal Fontaine, editor, *Automated Deduction – CADE 27*, volume 11716 of *Lecture Notes in Computer Science*, pages 538–549, Cham, 2019. Springer.
- [27] Tanel Tammet and Geoff Sutcliffe. Combining JSON-LD with first order logic. In *15th Int. Conf. on Semantic Computing (ICSC)*, pages 256–261. IEEE, 2021.
- [28] Lei Xu, Pierre Beckmann, Marco Valentino, and Andre Freitas. Adaptive LLM-symbolic reasoning via dynamic logical solver composition. In Vera Demberg, Kentaro Inui, and Lluís Marquez, editors, *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1187–1208, Rabat, Morocco, March 2026. Association for Computational Linguistics.
- [29] Yuan Yang, Siheng Xiong, Ali Payani, Ehsan Shareghi, and Faramarz Fekri. Harnessing the power of large language models for natural language to first-order logic translation (MALLS). In *Proceedings of ACL*, 2024.

- [30] Xi Ye, Qiaochu Chen, Isil Dillig, and Greg Durrett. SatLM: Satisfiability-aided language models using declarative prompting. In *NeurIPS*, 2023.