Proceedings of the Twelfth Annual Conference on Advances in Cognitive Systems ACS-2025 (261-276)

Automated Formalization of William Shakespeare's Venus and Adonis Using Large Language Models

Andrew S. Gordon GORDON@ICT.USC.EDU
Institute for Creative Technologies, University of Southern California, Los Angeles, CA 90094 USA

Abstract

Advances in large language models have created new opportunities for the translation of natural language into formal representations for use in symbolic reasoning systems, particularly when the formal vocabulary is well-specified with sufficient training examples. In this paper, we explore the more challenging problem of automating the knowledge engineering task of authoring the formal vocabulary itself, based on the representational needs of the input text. We describe a methodology that incrementally builds a formal vocabulary of first-order predicate-argument definitions and constant terms by prompting a large language model and restricting its output generations to adhere to tightly constrained formalisms. We evaluate this methodology in its application to a challenging literary text, William Shakespeare's narrative poem "Venus and Adonis" (1593), with a focus on how well new vocabulary is reused by the large language model in after it has been introduced. We then investigate the semantic content of the generated formalisms by manually sorting the vocabulary into a taxonomy of 48 foundational areas used in knowledge representation research. Our results point to several challenges in fully-automated knowledge engineering pipelines, but also point to new opportunities in using large language models to support axiomization of foundational theories.

1. Introduction

The phenomenal rise of large language models (LLMs) in the field of artificial intelligence has also challenged prevailing theories and methodologies in the adjacent fields of cognitive psychology and cognitive systems. While few would question the impressive abilities of LLMs, their success does not obviate the need for computational cognitive models that account for the enormous breadth of human reasoning skills, or the scientific pursuit of such models. Still, some of the enthusiasm for formal logic in models of automated reasoning has certainly waned over the years, with fewer researchers today pursuing approaches that involve the hand-authoring of knowledge base axioms of foundational commonsense theories. In the current climate, it is a fair question to ask: What is the role of logic-based formalisms in this new world of LLMs?

In this paper we flip this question around, and ask: What role can LLMs play in facilitating the use of logic-based formalisms in our computational models of cognitive systems? Arguably, the two core strengths of today's LLMs are in their command over natural language processing tasks and their enormous breadth of commonsense knowledge about the everyday life. As such, these systems seem well-suited for tackling two of the hardest problems in the practical use of formal logic in reasoning systems. First, we should expect that LLMs would be very good at converting natural lan-

guage expressions of facts or knowledge into logical form, as well as from logic back into language. Indeed, recent work has shown considerable success in LLM-based conversion of language into SQL queries and mathematical expressions, facilitating useful hybrid language-reasoning systems. Second, we should expect that LLMs would be good at articulating the content of core commonsense knowledge domains, with the right prompting. Their excellent performance on commonsense reasoning benchmarks provides clear indication that this commonsense knowledge is encoded, in latent form, among the weights of these models, and is readily employed in the next-word prediction task.

To explore these opportunities for LLM-assisted integration of logical reasoning, this paper investigates the use of an LLM in the formalization of a complex narrative text, namely William Shakespeare's lengthy narrative poem, Venus and Adonis, likely his first published work (1593). While previous research has demonstrated the use of LLMs in converting text into well-specified formalisms, here we investigate the more challenging knowledge engineering problem of authoring the formal vocabulary itself, based on the representational needs of the input text. After reviewing previous research, we describe a methodology that incrementally authors a representational vocabulary of predicate-argument literals in first-order logic, as well as constants for entities in the domain of discourse. New literal forms and constants are introduced by an LLM when it judges the existing vocabulary insufficient to encode the propositional content of an input text, where this propositional content is itself a distillation of the text performed by the LLM. We describe the application of this approach to the 199 six-line stanzas in Venus and Adonis, resulting in a representational vocabulary of 673 definitions of predicate-argument forms, along with 847 domain constants. To better understand the representation choices made by the LLM, we manually sorted the literal definitions into a taxonomy of 48 core commonsense representational areas resulting from previous (human-authored) knowledge engineering research.

2. Related work

Although some research in formal knowledge representation has been criticized for focusing excessively on simple domains, this is not true of previous work on the formalization of literary works. Hobbs (1990), in his book "Literature and Cognition," illustrates the difficulty of the knowledge representation problem in his analysis of several diverse texts, including a sonnet by John Milton and a novella by Gérard de Nerval. In applying his theory of discourse coherence to these texts, Hobbs argues that the process of discourse interpretation must juggle both local and global coherence, employing numerous high-level discourse relationships in order to identify the underlying coherence structure. Elsewhere, Hobbs et al. (1993) expanded on how complex discourse interpretation of this sort could be modeled in computers as logical abduction, where the logical form of an input text encodes a reader's best explanation of the words as observed, given the reader's knowledge of syntax, discourse, and the commonsense world. It would be many years before computational systems existed that implemented this proposal to any degree (Inoue et al., 2014), requiring major advances in statistical syntactic parsing (McClosky et al., 2006), fast logical abduction (Inoue & Inui, 2013), and the automated creation of sufficiently large knowledge bases (Ovchinnikova, 2012).

In many respects, these previous works toward the logical formalization of discourse structure stem from an older tradition of structuralism in narratology, e.g., Vladimir Propp's (1928) early dissection of the compositional plot structure in folktales. Finlayson (2016) demonstrated that the high-level narrative structures that Propp's identified (functions) could be inferred algorithmically from lower-level discourse structures (syntax, co-reference, semantic roles) in a hand-annotated corpus of fifteen folktales. However, the enormous challenges of curating a high-quality annotated corpus of this sort remains a significant obstacle in learning narrative structures at this level of abstraction.

Part of the appeal of LLMs in high-level narrative analysis is to reduce the need for hand-annotated training corpora, at least with regard to the annotation of the lower-level discourse structure. Huge stores of unannotated text, along with massive compute time, seem to be all that is necessary to implicitly learn enough of the essential discourse features to perform reasonably well on various discourse interpretation tasks in zero-shot or few-shot training contexts. Previous efforts to automatically translate natural language into structured formalisms include a two-step method called Semantic Parsing by Paraphrasing (SPP) (Berant & Liang, 2014; Shin et al., 2021). In this framework, the first step is to use a language model to paraphrase a natural language utterance into a canonical form, i.e., a text-to-text translation where the target paraphrase of the input is trivially converted into its formal form, in the second step, using a formal grammar or set of hand-authored rules. Recent advances in this approach have sought to eliminate the need to fine-tune large language models in the paraphrase step via clever sequences of engineered prompts (Yang et al., 2022) and improve compositional generalization by incorporating syntactical decomposition in prompt sequences (Drozdov et al., 2022).

A common feature of SPP and related approaches is that the target formalisms are all well defined in advance. For example, a typical experimental task in this line of work is to convert English queries into canonical query expressions in natural language, and then into the well-formed format of SQL database queries, e.g., using the Geo-Query dataset (Zelle & Mooney, 1996) of 800 SQL queries to a database of geographic statistics. Successful completion of this task requires that the output be valid SQL statements, using known table names (e.g., city, state) and standard SQL connectives (e.g., SELECT, WHERE). A key difference between previous work and our present investigation is the lack of a well-defined target formalism. Instead, our aim here is to investigate an LLM's ability to devise the necessary representational vocabulary, as needed, to complete the formalization task.

3. Methodology

This section describes a method for using an LLM to incrementally devise a logical vocabulary, as needed, to represent the logical form of an input text. While this method makes no assumptions about the structure of the input text, we formulate that the output of the process consists of three parts. First is a representation of the input text itself as an ordered list of ground literals in first-order logic, i.e., a predicate-argument structure where the arguments are all constants. Second is a vocabulary of predicate-argument forms, consisting of lifted first-order literals (where the arguments are all variables) and an English-language gloss that defines the relationship between predicated

arguments. Third is a vocabulary of constants in the domain of the input text, along with their English-language gloss.

Our choice to limit the output representations to an ordered list of ground literals does limit the structural complexity of the resulting logical form to that which emerges from shared constants across literals. However, we have found this style of representation sufficient to represent narratives in previous work (Gordon, 2018), particularly when viewed as intermediary representations for input to downstream interpretation processes.

3.1 Formalization Algorithm

The basic formalization algorithm used in this investigation is shown in Figure 1, a Python function that takes as input a text and any available initial representational vocabulary or constants, and returns as output the logical forms of the text along with updated representational vocabulary and constants. The six functions that are called via this algorithm each query a locally hosted LLM to generate a response, as follows.

```
def formalize_text(text, vocab, consts):
    result = []
    sentences = simple_sentences(text)
    for sentence in sentences:
        if not sufficient_vocab(sentence, vocab):
            vocab.append(acquire_vocab(sentence, vocab))
        if not sufficient_consts(sentence, consts):
            consts.extend(acquire_consts(sentence, consts))
        result.append(ground_literal(sentence, vocab, consts))
    return (result, vocab, consts)
```

Figure 1: Formalization Algorithm

The function `simple_sentences` prompts the LLM to process the input text into a list of simple sentences that can be expressed as a single positive logical literal. The algorithm then iterates over these simple sentences. For each sentence, `sufficient_vocab` first asks the LLM to judge whether the existing logical vocabulary is sufficiently expressive to capture the meaning of the sentence. If not, `acquire_vocab` prompts the LLM to create a new vocabulary definition for a predicate-argument form, as a lifted literal in first-order logic. Next, `sufficient_consts` asks the LLM to judge whether the existing catalog of constants is sufficient to refer to the entities in the sentence, and if not, `acquire_consts` prompts the LLM for one or more additions. Finally, `ground_literal` prompts LLM to use the (possibly expanded) formal vocabulary and constants to represent the sentence as a ground literal, which is added to the result.

Importantly, none of these six functions allow the LLM to generate totally unconstrained text responses. In each case, we use functionality of the locally hosted LLM software (`llama.cpp`)to constrain the generated outputs to conform to a given JSON schema, as described in the next sec-

tions. In this manner, we are able to guarantee that the responses from each of these six functions are valid sentence lists, booleans, vocabulary definitions, and constant definitions, accordingly.

3.2 Prompting for Simple Sentences

The first LLM task in this methodology is to rewrite the input text into a set of simpler sentences. Table 1 shows the prompt template that was used for this task, where *{text}* is replaced with all or successive portions of the entire text to be formalized, e.g. one paragraph at a time. The aim of this step is to reduce the complexity of the input text so that the corresponding simple sentences can each be represented using a single literal in first-order logic, i.e., using a single predicate form. By requesting the removal of subordinate clauses and co-references, the intention is to enable each simple sentence to be formalized independently of others in the set.

In this and all subsequent functions, the LLM's generation is constrained by passing a JSON schema of acceptable output. In this function, this schema is simply a list of strings of any length.

Table 1: Prompt Template for `simple_sentences`

Your job is to read an input statement and rewrite it as a set of simpler sentences that do not include subordinate clauses or pronoun references. Each output sentence should be easy to read and stand on its own, without referencing any of the other sentences. The format of your output should be a JSON array of strings. Here is the input sentence: {text}

3.3 Prompting for Vocabulary and Constants

The next four tasks in this methodology are to determine if the vocabulary and constants are sufficient to adequately represent each of the simple sentences derived from the input text. Table 2 shows the two prompt templates related to the sufficiency of the vocabulary. In both prompts, the entire current representational vocabulary is inserted in place of the *{vocab}* tag as a JSON string representation of a list of available predicates, their variable arguments, and an English gloss that defines the relationship among variables. In the first prompt, the output is constrained using a JSON schema for the enumerated strings "yes" and "no". For the second prompt, the JSON schema is for a lifted literal of exactly the form of current vocabulary entries.

The two corresponding prompt templates for constants are omitted for space, but follow an almost identical format. The existing constants are inserted as a JSON string representation of a list of available constant terms, along with an English gloss that defines their reference. Likewise, the LLM generation to `acquire_consts` is constrained using a JSON schema for a list of one or more new constants in exactly this form.

3.4 Prompting for Output Literals

Having established that there are sufficient representational vocabulary and constants to formalize a given simple sentence, the final LLM task is to represent the simple sentence as a ground literal. Table 3 shows the prompt template used for this task, which again includes all available literal

Table 2: Prompt Templates for `sufficient_vocab` and `acquire_vocab`

function	prompt template
`sufficient_vocab`	*Your job is to answer a difficult yes/no question. The question is this: given a simple english sentence and vocabulary of logical forms in first-order logic, is there some item in the vocabulary that is expressive enough to represent the meaning of the sentence? To be sufficiently expressive, there must be some logical form in the vocabulary that has a predicate that captures the relationships between the entities referenced in the english sentence. If there is some vocabulary item in the list that is sufficient, then answer "yes". If there is not an existing vocabulary entry that is sufficient to represent the sentence, then answer "no". Here is the list of vocabulary items: {vocab} ...and here is the sentence that need to be represented: {sentence}. Please provide your response in one word, either "yes" or "no".*
`acquire_vocab`	*Your job is to add a new entry to an existing vocabulary of first-order logical forms. Each entry in the vocabulary has a predicate and a list of arguments (variables) along with an English gloss that describes how the arguments relate to each other. Here is the vocabulary as it exists before your addition: {vocab} ...Unfortunately, this list is not sufficiently expressive to represent the following sentence: {sentence} Your job is to write a single addition to this vocabulary that enables this sentence to be represented. Please format your response as a JSON object with a "predicate", "arguments" consisting of an array of variables, and a "gloss" which briefly explains the relationship between the variables.*

forms and domain constants represented as JSON strings. Here the output generation is constrained using a JSON schema for a ground literal, which is the same as a lifted literal, above, except that arguments must be constants. In this prompt, the LLM is allowed to restate the input sentence as the English gloss of the ground literal, and in our observations it nearly always takes that option.

Table 3: Prompt Templates for `ground_literal`

Your job is to represent a simple english sentence as a first-order logical literal using only logical forms from fixed vocabulary and using constants from a fixed list. The vocabulary and the list of constants is sufficient to adequately represent the sentence as a single literal; you just need to pick the right logical form and select the right constants and put them in the right place in the argument list. Here is the vocabulary of logical forms that you have to choose from: {vocab} ...and here is list of constants that you can use for arguments: {consts} ...and here is the sentence that you need to represent: {sentence}. Please format your response as a JSON object with a "predicate", a list of "arguments" (constants), and an english gloss. You can repeat the original sentence for use as the english gloss.

3.5 Prompt Engineering

The six prompt templates described in this section were authored without extensive prompt engineering, and are largely first drafts of the textual instructions we provided to the LLM. In composing these templates, our approach was to verify that the instructions could be understood by an LLM in a typical chat interface, without constraining the model's generation using JSON schemas. In all cases we observed that the LLM understood the instructions and could perform the tasks despite our use of technical terms related to grammar and formal logic. In these unconstrained generations we observed several variations in how the LLM formatted the predicates and arguments of logical literals, as well as verbose justifications of these responses. We expect that these templates could be improved in several ways, such as the inclusion of examples of how the LLM should respond in various cases for each subtask. In future work, a more rigorous prompt-engineering methodology could be pursued with the development of automated benchmarking tools for each subtask.

4. Application to Shakespeare's Venus and Adonis

We investigated the application of our formalization methodology to a particularly challenging literary text, William Shakespeare's narrative poem *Venus and Adonis*. Thought to be Shakespeare's first published work (1593), the poem reimagines the Latin poet Ovid's telling of the meeting of the Roman goddess of love and Adonis, an extremely handsome mortal man. In Shakespeare's poem, Venus sees and falls in love with Adonis, and comes down to earth to seduce him. Adonis, however, is a man uninterested in love - even the love of a goddess - and would prefer to spend his time hunting. After Adonis refuses her and scorns her for her attempts, Venus faints. Fearing that he may have killed her, Adonis comforts her and gives in to her requests for a kiss, which further encourages her. Adonis refuses her request to see each other again the next day, as he will be hunting wild boar, leading Venus to warn him of her vision that he will be killed by a boar. She again attacks him in lust, and Adonis responds by lecturing her on the difference between love and lust, before leaving her in tears. The next day, Venus indeed finds Adonis killed in the woods by a wild boar. The devastation of her loss causes her to decree, as the goddess of love, that love will henceforce be mixed with suspicion, fear, and sadness. She then departs in her own sadness, sequestering herself in bereavement to Paphos on Cyprus.

Shakespeare narrates the story of Venus and Adonis as a poem consisting of 199 stanzas, each as six lines of iambic pentameter rhyming ABABCC, for a total of 1,194 lines. To process the full text using the algorithm shown in Figure 1, we first divided the text into individual six-line stanzas. The stanzas were then iteratively presented as the input text to the algorithm, with the output expanded vocabulary and constants used as inputs in each subsequent iteration.

In this investigation, we used Google's open-weight Gemma 3 large language model (Gemma Team, Google DeepMind, 2025), specifically the 27B-parameter variant with 4-bit K-M quantized weights in GGUF format. This model was hosted on a 2023 Mac Studio with an M2 Max processor with 96 GB unified memory using a `llama.cpp` local server, prompted from our custom Python scripts using the Python Requests module. We set the context window to 100K tokens, and disabled the model's Sliding Window Attention to better manage memory usage.

Table 4 shows the results of applying the `simple_sentences` function to the first stanza of Venus and Adonis, converting its six lines into nine simple sentences. This degree of expansion when rewriting the input text was consistent across all 199 stanzas, with a mean number of simple sentences for each six-line stanza of 9.13 (SD=2.14), for a total of 1,817 simple sentences.

Table 4: Example input text conversion into simple sentences

EVEN as the sun with purple-colour'd face	The sun had a purple face.
Had ta'en his last leave of the weeping morn,	The sun finished rising.
Rose-cheek'd Adonis hied him to the chase;	Adonis went to the hunt quickly.
Hunting he loved, but love he laugh'd to scorn;	Adonis loved hunting.
Sick-thoughted Venus makes amain unto him,	Adonis did not believe in love.
And like a bold-faced suitor 'gins to woo him.	Venus was feeling unwell.
	Venus went to Adonis with purpose.
	Venus began to try to win Adonis's affection.
	Venus acted like a confident suitor.

In considering the available expressivity of the representational vocabulary (initially empty), the LLM judged that eight of the nine simple sentences in Table 4 required the introduction of a new predicate-argument literal form. Table 5 shows the eight new vocabulary definitions that were added by the `acquire_vocab` function in our algorithm.

Table 5: Examples of the vocabulary added by `acquire_vocab`

predicate	arguments	gloss
has_face_color	?x, ?y	Entity ?x has a face color of ?c.
finished	?x	Event ?x is completed.
went_to	?x, ?y, ?m	Entity ?x went to location ?y using mode ?m.
loves	?x, ?y	Entity ?x loves activity ?y.
believes_in	?x, ?y	Entity ?x believes in entity ?y.
feeling	?x, ?s	Entity ?x is feeling state ?s.
attempt	?x, ?y, ?z	Entity ?x attempts activity ?y to achieve ?z.
acted_like	?x, ?y	Entity ?x acted like ?y.

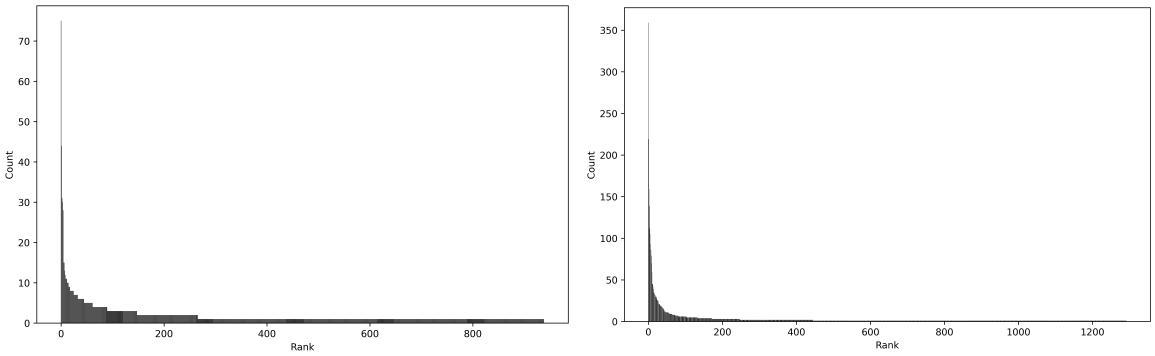
Likewise, the `acquire_consts` function introduced six new domain constants for use in the output literals, including the constants ADONIS and VENUS for the two main characters. Table 6 shows an example of the output of the `ground_literal` function, where the nine simple sentences from Table 4 are formalized using the eight new logical forms and six new domain constants.

Even in these first nine ground literals in Table 6 there are some notable representational choices. The efficient reuse of the predicate `went_to` is encouraging, but ignoring the negation when applying the `believe_in` form is somewhat concerning. Notable as well is the reuse of the constant HUNT to refer both to Adonis's favorite activity as well as to Venus's plan to seduce Adonis.

After processing all 1,817 simple sentences into an equal number of ground literals, our formalization algorithm had grown the size of the vocabulary to 939 literal forms and 1,292 domain

Table 6: Example formalisms generated by `ground_literal`

The sun had a purple face.	(has_face_color SUN PURPLE)
The sun finished rising.	(finished SUN)
Adonis went to the hunt quickly.	(went_to ADONIS HUNT QUICKLY)
Adonis loved hunting.	(loves ADONIS HUNT)
Adonis did not believe in love.	(believes_in ADONIS LOVE)
Venus was feeling unwell.	(feeling VENUS UNWELL)
Venus went to Adonis with purpose.	(went_to VENUS ADONIS PURPOSE)
Venus began to try to win Adonis's affection.	(attempt VENUS HUNT ADONIS)
Venus acted like a confident suitor.	(acted_like VENUS SUITOR)



(a) Literal forms

(b) Constants

Figure 2: Histograms of the frequency of reuse.

constants. To explore further how literal forms and constants were reused during the execution of this formalization process, we tabulated the usage of each form and constant across all 1,817 ground literals. The results, plotted as histograms in Figure 2, show that the majority of literal forms (71.67%) and constants (65.56%) were used only once across all ground literals. For literal forms, the top 10 most-used predicates are as follows: is (75), says_is (44), says (31), would (30), is_like (28), wants (15), do_not (15), feel_emotion (13), cries (12), and loves (11). These usage rates for predicates seem reasonable, given that the poem Venus and Adonis primarily focuses on tearful conversations between two characters about love. Likewise, the top 10 most-used constants are as follows: SHE (359), HE (219), YOU (159), I (139), TIME (112), LOVE (105), HIS (93), HER (86), ACTIVITY (79), and HIM (70). These usage rates for constants, excessively dominated by pronouns, indicate that our approach largely failed to reuse constants for VENUS and ADONIS as the process progressed.

5. Comparison to Foundational Theories

The quantitative findings reported in the previous section provide some insight into the ability of an LLM to invent and reuse formal vocabulary. However, some important questions remain. These results show some reuse of literal forms, but assessing whether this rate is high or low depends on their semantic scope. If an analysis of these forms reveals substantial semantic overlap or trivial variations, then these rates are lower than would be desired. If instead the semantic scope of these forms is comparable to formalizations of knowledge domains produced by professional knowledge engineers or artificial intelligence researchers, then we would have a more favorable opinion of the LLM's knowledge engineering abilities and reuse rates.

In order to conduct a qualitative analysis of the LLM's invented representational vocabulary, we manually sorted each of the 939 LLM-authored lifted literal definitions into an existing taxonomy of foundational and commonsense knowledge domains. As a target taxonomy, we selected the 48 foundational representational areas identified by Gordon (2004) in his large-scale analysis of 372 high-level planning strategies across 10 planning domains, e.g., the warfare strategies in Sun Tzu's Art of War and the political strategies of Niccolo Machiavelli's The Prince. This taxonomy was selected in part due to recent interest in its use as a basis for comprehensive benchmarks in automated commonsense reasoning evaluations (Santos et al., 2022; Kejriwal et al., 2022), but also because a large portion of this taxonomy was subsequently formalized as a theory of commonsense psychology in first-order logic (Gordon & Hobbs, 2017). This allowed us to draw some coarse-grained comparisons between the LLM-authored vocabulary and a broad-coverage taxonomy, as well as fine-grained comparisons between the representational choices made by the LLM and by leaders in the field of knowledge representation.

Table 7 shows the distribution of LLM-authored literal forms after manually sorting them into the taxonomy's 48 representational areas. From this coarse analysis, it is evident that the LLM-authored formalisms, in general, are widely distributed across most foundational areas identified in previous large-scale knowledge engineering research. It is notable that the representational areas that specifically involve direct action among agents or the physical world (areas 18, 19, 20, 21, 22, and 48) cover a substantial portion of all LLM-authored formalisms, possibly owing to the nature of Venus and Adonis as a narrative of the interaction between two people. The absence of any forms related to Plans, Planning Modalities, Plan Adaptation, and Scheduling suggests that planning beyond the present moment is not a major concern among these two characters, while concerns for States, Events, Space and Physical Entities are more important in this narrative.

In reviewing the individual sets of forms sorted into these 48 representational areas, we do not find evidence for widespread semantic overlap among LLM-authored forms. In a handful of cases there are nearly-identical forms introduced, e.g., (amazed_by ?x ?y) and (is_amazed_by ?x ?y) in the area of 31 (Managing Expectations). More often, we find that representational areas with an over-abundance of LLM-authored forms include predicates that would be more elegantly split into two different literal forms. For example, the 37 LLM-authored predicates related to area 24 (Similarity Comparisons) include predicates that assert similarity between entities in a specific characteristic, such as glisters_like and consumes_like, or that judge one entity greater than another along some specific dimension, such as fairer_than and superior_to. The formalization algorithm we investigated in this research does not permit multiple literals for a single simple sentence, however.

Table 7: Distribution of 939 LLM-Authored Literal Forms Across Representational Areas

01. Time (8)	25. Memory retrieval (3)
02. States (46)	26. Emotions (34)
03. Events (43)	27. Explanations (7)
04. Space (45)	28. World envisionment (35)
05. Physical entities (52)	29. Execution envisionment (35)
06. Values and quantities (11)	30. Causes of failure (4)
07. Classes and instances (3)	31. Managing expectations (7)
08. Sets (0)	32. Other agent reasoning (9)
09. Agents (12)	33. Threat detection (3)
10. Agent relationships (17)	34. Goal management (10)
11. Communities and organizations (5)	35. Planning modalities (0)
12. Goals (20)	36. Planning goals (11)
13. Goal themes (4)	37. Plan construction (2)
14. Plans (0)	38. Plan adaptation (0)
15. Plan elements (3)	39. Design (6)
16. Resources (17)	40. Decisions (4)
17. Abilities (14)	41. Scheduling (0)
18. Activities (49)	42. Monitoring (14)
19. Communication acts (66)	43. Execution modalities (43)
20. Information acts (16)	44. Execution control (16)
21. Agent Interaction (57)	45. Repetitive execution (2)
22. Physical interaction (56)	46. Plan following (6)
23. Managing knowledge (34)	47. Observation of execution (8)
24. Similarity comparisons (37)	48. Body interaction (33)

For the representational areas related to commonsense psychology, we were able to conduct a more fine-grained comparison between the LLM-authored literal forms and those authored by Gordon and Hobbs (2017) as part of their methodical formalization of these domains. Specifically, we compared the definitions of predicate forms that are listed at the end of each chapter in Gordon and Hobbs's theory with LLM-authored literal forms sorted into the corresponding category. Table 8 shows an example of a fine-grained comparison in the representational area of "Design", area 39 in Table 7 and Chapter 40 in Gordon and Hobbs formalization.

Table 8 indicates that in some cases, the LLM-authored literal form is nearly identical in semantic scope to one in a well-crafted formal theory, e.g., designed_to overlaps strongly with planFor, and is_for strongly overlaps with intendedUse. In other cases, forms differ only slightly in scope, e.g., has_flaw refers to a characteristic of an instance, while flawedDesign refers to all instances in class. As seen elsewhere, the LLM-authored literals include some that would have been better to divide into two forms, e.g., should_not_fade and is_for_bearing. From a knowledge engineering perspective, the most interesting LLM-authored literal is possibly is_simple, which appears out-

Table 8: Comparison of Literal Forms in the Representational Area of Design

author	form	gloss
LLM	(has_flaw ?x ?y)	Entity ?x has flaw ?y.
	(is_for ?x ?y)	Entity ?x is intended for activity ?y.
	(is_for_bearing ?x)	Entity ?x is intended for bearing.
	(is_simple ?x ?s)	Entity ?x is simplicity level ?s.
	(should_not_fade ?x ?c)	Entity ?x should not fade in color ?c.
	(designed_to ?x ?y ?z)	Entity ?x is designed to cause state ?z to entity ?y.
Human	(artifact ?x)	?x is an artifact.
	(planFor ?p ?x)	?p is the plan realized in the structure of artifact ?x.
	(artifact1 ?x)	?x is a physical artifact.
	(intendedUse ?e1 ?x ?a)	?e1 is the intended use of artifact ?x by agent ?a.
	(design ?x)	?x is a design.
	(notAchieve ?s ?e)	The occurrence of the eventualities in ?s do not result in ?e occurring.
	(terminalSubgoalsOf ?s ?p)	The eventualities in ?s are the terminal subgoals in plan ?p.
	(flawedDesign ?x)	?x is a flawed design.
	(designing ?e ?a ?x ?s)	?e is a designing activity by agent ?a in designing ?x via a sequence ?s of designs.
	(designConstraint ?e ?x)	?e is a design constraint design ?x satisfies.
	(designAdaptation ?e ?a ?x)	The designing activity ?e by agent ?a of design ?x is a planning activity by adaptation.
	(designFailure ?e ?a)	?e is a designing activity by agent ?a that fails.
	(moreAdherentToDesign ?x1 ?x2 ?x)	Instance ?x1 adheres more closely to design ?x than instance ?x2 does.

side the representational scope of the hand-crafted theory, encouraging the further expansion of the hand-authored formalization toward concerns for simplicity and elegance as qualities of designs.

6. Discussion

Our investigation of this formalization methodology yielded some encouraging results. Through the quantitative and qualitative analysis of its application to Shakespeare's Venus and Adonis, we found that the LLM was indeed fairly competent at inventing new formal vocabulary when needed, and subsequently reusing this vocabulary in latter representations. The semantic breadth of the LLM-authored vocabulary was comparable in scope to established broad-coverage taxonomies, and

the specific literal forms were often comparable to those of skilled knowledge engineers. Although it must be noted that defining literal forms is only a small part of axiomatizing a foundational domain, the modest success of our formalization algorithm suggests that fully-automated domain axiomization of foundational domains may be possible with future research progress. We also see some role of our current algorithm in supporting human knowledge engineering in a mixed-initiative fashion. For example, we imagine that this algorithm could be useful in the early stages of axiomitizing knowledge in specific technical or professional domains, where selected domain-specific texts could be processed in order to establish the required scope of the knowledge engineer's axiomization task. Likewise, our algorithm's capacity for reusing existing vocabulary may work equally well with hand-crafted definitions, creating opportunities to evaluate and refine axioms of domain theories with natural language benchmarks.

This investigation also highlighted many shortcomings in our proposed methodology, many of which stem directly from the design of the formalization algorithm itself, in Figure 1. Here, interaction with the LLM is cast as a set of six functions, independent of each other, each of which prompts the LLM with an empty context. As such, this algorithm is horribly inefficient. In the worst case, the algorithm passes the entire vocabulary and entire list of constants to the LLM three times each - for every simple sentence derived from the input text. In the early stages of processing, when the vocabulary is small, this inefficiency is negligible. However, after processing 1,817 simple sentences in Venus and Adonis, the vocabulary had grown to 939 literal forms and 1,292 domain constants, such that processing the latter stanzas each took several *hours* of compute time on our local hardware. While some speed improvements might be made by tuning the configuration of the local `llama.cpp` server, particularly the memory management of the k-v cache, a better solution would be to redesign the formalization algorithm altogether. Instead of six independent functions, the tasks directed to the LLM would be better conceived as dialogue moves in an ongoing conversation, in the same way that people interact with LLM providers via web chat interfaces. Ideally, the formalization is conducted incrementally over the course of a single chat that fits within the context window of the LLM, such that all of the newly invented literal forms and constant definitions are still in context and available for use later in the conversation. Such a conversational approach should also improve the LLM's ability to reuse constants for named entities, e.g., VENUS and ADONIS, rather than pronouns, as the larger narrative context should make it easier for the LLM to resolve the references when deriving the simple sentences.

A further improvement on our methodology would be to enable simple sentences to be formalized using multiple ground literals. Constraining the LLM to single predicate-argument form led to the introduction of spurious predicates that were used only once during processing, e.g., the fairer_than and consumes_like predicates discussed above. For example, the first simple sentence in Venus and Adonis, "The sun had a purple face," might be more elegantly represented with a combination of has_part and has_color literals and a FACE constant, rather than awkward formulation seen in Table 6. More nuanced relationships between ground literals could also be supported by encouraging the use the so-called *eventuality notation* as seen in some of the human-authored definitions in Table 8, e.g., in intended_use and not_achieve. These representational choices are often associated with the particular styles of certain well-known knowledge engineering researchers, and it may be possible to mimic these styles in LLMs with careful prompting with illustrative examples.

7. Conclusions

In the current era of rapid progress in LLMs it is temping, for many, to see the technologies of artificial intelligence as divided into those of the past versus those of the present. First-order logic, with its roots in the 18th and 19th-century work of philosophers including Gottlob Frege, Bertrand Russell, and Ludwig Wittgenstein, certainly would be cast into the pile of the past. This somewhat divisive view persists despite the pervasive logical foundations of nearly all of contemporary computational infrastructure, from relational databases, boolean satisfiability solvers, and programming language compilers, to the very cores of our CPUs. Instead, it may be more productive to view LLMs as a powerful new addition to the suite of established tools available to us across the computational sciences, including cognitive systems research. As with other tools in this suite, many advances can be realized by combining the strengths of different tools in novel ways.

In this paper, we investigate the role that LLMs might play in facilitating the use of logic-based formalisms in our computational models of cognitive systems. The contribution of this paper is to advance an LLM-enabled methodology for generating first-order logic representations of complex narrative texts, while inventing the formal vocabulary used in these representations incrementally as needed. Our selected text, William Shakespeare's Venus and Adonis, further demonstrates that this method is applicable to difficult and lengthy literary works, yielding a formal vocabulary that veers in scope toward the thematic concerns of the source text. Our quantitative and qualitative analyses show that the method is successful in reusing formalisms invented earlier in the analysis. By sorting the LLM-authored vocabulary into an established taxonomy of foundational representational areas, we see substantial overlap in the conceptual breadth of representations. By comparing LLM-authored vocabulary with well-crafted formalisms in areas related to commonsense psychology, we see some similarity in the representational choices that are made. This investigation also highlights some of the limitations of our method, particularly with respect to its inefficient use of independent and decontextualized functions, and the lack of multi-literal representations of simple sentences. To address these limitations, we proposed new directions for future research on a dialogue-oriented formalization methodology, along with enhanced prompting that encourages the more expressive and stylistic representational choices of knowledge engineering researchers.

We view this investigation as a further step toward fully-automated logical formalization of both foundational theories and specialized domain theories, to include the axiomatic definitions and inference rules necessary to support automated (logical) reasoning of various sorts. With LLMs working to bridge the gap between natural language and logical formalisms, we also see potential applications in the future where logic-based reasoning, e.g., automated deduction and abduction, is more seamlessly integrated into the dialogue interfaces and agentic workflows that are pervasive in today's applications of LLMs.

Acknowledgements

Research was sponsored by the Army Research Office and was accomplished under Cooperative Agreement Number W911NF-25-2-0040. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Office or the U.S. Government. The U.S. Government

References

Berant, J., & Liang, P. (2014). Semantic parsing via paraphrasing. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 1415–1425). Baltimore, Maryland: Association for Computational Linguistics.

Drozdov, A., Scharli, N., Akyuurek, E., Scales, N., Song, X., Chen, X., Bousquet, O., & Zhou, D. (2022). Compositional semantic parsing with large language models. *ArXiv*, *abs/2209.15003*.

Finlayson, M. A. (2016). Inferring propp's functions from semantically annotated text. *The Journal of American Folklore*, *129*, 55–77.

Gemma Team, Google DeepMind (2025). Gemma 3 technical report.

Gordon, A. S. (2004). *Strategy representation: An analysis of planning knowledge*. Lawrence Erlbaum Associates.

Gordon, A. S. (2018). Interpretation of the Heider-Simmel film using incremental etcetera abduction. *Advances in Cognitive Systems*, *6*, 1–16.

Gordon, A. S., & Hobbs, J. R. (2017). *A formal theory of commonsense psychology: How people think people think*. Cambridge, UK: Cambridge University Press.

Hobbs, J. R. (1990). *Literature and Cognition*, volume 11 of *Lecture Notes*. Center for the Study of Language and Information.

Hobbs, J. R., Stickel, M. E., Appelt, D. E., & Martin, P. (1993). Interpretation as abduction. *Artificial Intelligence*, *63*, 69–142.

Inoue, N., & Inui, K. (2013). Ilp-based inference for cost-based abduction on first-order predicate logic. *Journal of Natural Language Processing*, *20*, 629–656.

Inoue, N., Ovchinnikova, E., Inui, K., & Hobbs, J. (2014). *Weighted abduction for discourse processing based on integer linear programming*, (pp. 33–55). United States: Elsevier Inc.

Kejriwal, M., Santos, H., Mulvehill, A., et al. (2022). Designing a strong test for measuring true common-sense reasoning. *Nature Machine Intelligence*, *4*, 318–322.

McClosky, D., Charniak, E., & Johnson, M. (2006). Effective self-training for parsing. *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference* (pp. 152–159). New York City, USA: Association for Computational Linguistics.

Ovchinnikova, E. (2012). *Integration of world knowledge for natural language understanding*. Atlantis Thinking Machines. Atlantis Press.

Propp, V. (1928). *Morphology of the folktale: Second edition*. University of Texas Press.

Santos, H., Shen, K., Mulvehill, A. M., Razeghi, Y., McGuinness, D. L., & Kejriwal, M. (2022). A theoretically grounded benchmark for evaluating machine commonsense.

Shin, R., et al. (2021). Constrained language models yield few-shot semantic parsers. *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing* (pp. 7699–7715). Online and Punta Cana, Dominican Republic: Association for Computational Linguistics.

Yang, J., Jiang, H., Yin, Q., Zhang, D., Yin, B., & Yang, D. (2022). SEQZERO: Few-shot compositional semantic parsing with sequential prompts and zero-shot models. *Proceedings of NAACL'22 Findings*. Association for Computational Linguistics.

Zelle, J. M., & Mooney, R. J. (1996). Learning to parse database queries using inductive logic programming. *Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 2* (p. 1050–1055). AAAI Press.