Towards Large Language Model Architectures for Knowledge Acquisition and Strategy Synthesis

Paolo Giorgini, Andrea Mazzullo, Marco Robol and Marco Roveri

University of Trento

Abstract

To address the bottlenecks of knowledge acquisition and strategy synthesis, in the development of autonomous AI agents capable of reasoning and planning about dynamic environments, we propose an architecture that combines large language model (LLM) functionalities with formal verification modules. Concerning knowledge acquisition, we focus on the problem of learning description logic concepts to separate data instances, whereas, in a process mining setting, we propose to leverage LLMs to extract linear temporal logic specifications from event logs. Finally, in a strategy synthesis context, we illustrate how LLMs can be employed to address realisability problems in linear temporal logic on finite traces.

Keywords

Large Language Models, Knowledge Acquisition, Strategy Synthesis, Learning from Examples, Description Logics, Linear Temporal Logic

1. Introduction

The combination of machine learning methods, based on stochastic black-box architectures, with logic-based techniques, symbolic and explainable in nature, is considered of critical importance for developing AI-based autonomous agents that can evolve strategies and plans, or reason about their surroundings in the presence of newly acquired information [1, 2, 3, 4]. In this direction, the integration of large language models (LLMs) with knowledge representation features is receiving significant attention in the literature [5, 6, 7, 8, 9]. One approach aims at combining LLMs, which are known to perform well in natural language generation tasks, with integrated reasoning modules, used to address and solve formal problems in a provably correct way [10, 11, 12]. Another area that is recently gaining traction is concerned with the enhancement of LLMs with planning capabilities, to perform explainable scheduling tasks [13, 14, 15, 16, 17, 18].

When dealing with knowledge-intensive structured domains, or in the presence of data evolving over time in dynamic environments, autonomous AI agents face the following important challenges: (1) knowledge acquisition, that is, the task of extracting structured information from raw data in a given domain, in turn allowing for domain-specific or time-dependent conceptual modelling and reasoning; (2) strategy synthesis, i.e., the task of devising sequences of actions, possibly in response to environmental conditions or other agents’ choices, in order to reach a
Towards LLM-assisted generation of separating concepts, we propose the following architecture, 

\[ \text{version, and assume familiarity with DL concepts; see [35] for detailed preliminaries}. \]

First, 

\[ \text{addressed by concrete tool implementations for separating concept generation [50, 51, 52, 53].} \]

\[ \text{has been investigated both as a decision problem, from a theoretical perspective [31, 35], and} \]

\[ \text{requires that } \mathcal{K} = (\mathcal{O}, \mathcal{D}) \text{ be an } \mathcal{L} \text{ knowledge base, containing both the background axioms in} \]

\[ \text{the ontology } \mathcal{O}, \text{ as well as (ground) facts stored in the dataset } \mathcal{D}. \]

\[ \text{The positive individuals, } P, \text{ and negative individuals, } N, \text{ are subsets of ind(} \mathcal{D} \text{), i.e., of the individuals occurring in } \mathcal{D}. \]

\[ \text{The (weak) concept separability problem asks the following: given } \mathcal{K}, P, \text{ and } N, \text{ is there an } \mathcal{L} \text{ concept} \]

\[ \text{that separates the positive from the negative examples? That is: } \mathcal{K} \models C(e^+), \text{ for every } e^+ \in P; \text{ and } \mathcal{K} \not\models C(e^-), \text{ for every } e^- \in N. \text{(A strong version of the separability problem requires that } \mathcal{K} \models \neg C(e^-), \text{ for every } e^- \in N: \text{ we omit it for space reasons).} \]

\[ \text{This problem has been investigated both as a decision problem, from a theoretical perspective [31, 35], and addressed by concrete tool implementations for separating concept generation [50, 51, 52, 53].} \]

Towards LLM-assisted generation of separating concepts, we propose the following architecture, illustrated in the box below and summarised in Fig. 1 (left).

\[ \text{Input } (\mathcal{K}, P, N). \]

\[ \text{Output Separating } \mathcal{L}\text{-concept } C \text{ for } (\mathcal{K}, P, N), \text{ if it exists.} \]

\[ \text{Procedure} \]

\[ \text{• Prompt input } (\mathcal{K}, P, N) \text{ to the LLM-based DL concept generation module.} \]

\[ \text{• While no separating concept is found, repeat the following steps:} \]

\[ \text{– ask the LLM module to candidate a separating } \mathcal{L} \text{ concept } C; \]

\[ \text{– check with a DL } \mathcal{L} \text{ reasoner if } \mathcal{K} \models C(e^+), \text{ for all } e^+ \in P; \text{ and } \mathcal{K} \not\models C(e^-), \text{ for all } e^- \in N:} \]

\[ \text{• if a counterexample } \bar{e} \text{ is found, i.e., } \mathcal{K} \not\models C(\bar{e}), \text{ with } \bar{e} \in P, \text{ or } \mathcal{K} \models C(\bar{e}), \text{ with } \bar{e} \in N, \text{ pinpoint } \bar{e} \text{ to the LLM module;} \]

\[ \text{• otherwise, return } C \text{ as separating concept.} \]

\[ \text{Process Mining in LTL. A challenge in process mining [54, 55] consists in the identification,} \]

\[ \text{given goal, for automatic programming and planning purposes. From a foundational viewpoint,} \]

\[ \text{two formalisms can be arguably considered sufficiently expressive for these problems: description} \]

\[ \text{logics (DLs), a well-known family of knowledge representation languages devised to be} \]

\[ \text{computationally well-behaved [19, 20]; and linear temporal logic (LTL), which extends classical} \]

\[ \text{propositional logic with time modalities interpreted on linear structures [21, 22], and is widely} \]

\[ \text{applied in computer science and AI [23, 24, 25, 26, 27, 28].} \]

\[ \text{By relying on these formalisms, we propose an integrative AI architecture based on LLMs} \]

\[ \text{to address both knowledge acquisition and strategy synthesis tasks. We first illustrate, in} \]

\[ \text{Section 2, our framework within the knowledge extraction context. This approach is related} \]

\[ \text{to: ontology and concept learning or separability in DLs [29, 30, 31, 32, 33, 34, 35], as well as} \]

\[ \text{reverse engineering of formulas in LTL [36, 37, 38, 39]; problems in inductive (and abductive) } \]

\[ \text{reasoning [40, 41]; the model of active learning with membership queries in machine learning [42,} \]

\[ \text{43, 44, 45]; and the query-by-example approach from database theory [46].} \]

\[ \text{In Section 3, we present our architecture for the strategy synthesis setting. This shares connections with LLM-based planning} \]

\[ \text{approaches [47, 48], and to counterexample-guided inductive synthesis [49] in the field of automatic} \]

\[ \text{programming. We briefly discuss in Section 4 future research directions.} \]

\[ \text{2. LLM-Driven Knowledge Acquisition} \]

\[ \text{Concept Learning in DLs. In DLs, concept learning is the task of automatically generating,} \]

\[ \text{from a set of examples, a concept description that correctly represents them (see also [35] and} \]

\[ \text{references therein). Related to this question, for a given DL language, the following concept} \]

\[ \text{separability problem has been investigated (for space limitations, we present here a simplified} \]

\[ \text{version, and assume familiarity with DL concepts; see [35] for detailed preliminaries). First,} \]

\[ \text{given a DL } \mathcal{L}, \text{ let } \mathcal{K} = (\mathcal{O}, \mathcal{D}) \text{ be an } \mathcal{L} \text{ knowledge base, containing both the background axioms in} \]

\[ \text{the ontology } \mathcal{O}, \text{ as well as (ground) facts stored in the dataset } \mathcal{D}. \]

\[ \text{The positive individuals, } P, \text{ and negative individuals, } N, \text{ are subsets of ind(} \mathcal{D} \text{), i.e., of the individuals occurring in } \mathcal{D}. \]

\[ \text{The (weak) concept separability problem asks the following: given } \mathcal{K}, P, \text{ and } N, \text{ is there an } \mathcal{L} \text{ concept} \]

\[ \text{that separates the positive from the negative examples? That is: } \mathcal{K} \models C(e^+), \text{ for every } e^+ \in P; \text{ and } \mathcal{K} \not\models C(e^-), \text{ for every } e^- \in N. \text{(A strong version of the separability problem requires that } \mathcal{K} \models \neg C(e^-), \text{ for every } e^- \in N: \text{ we omit it for space reasons).} \]

\[ \text{This problem has been investigated both as a decision problem, from a theoretical perspective [31, 35], and} \]

\[ \text{addressed by concrete tool implementations for separating concept generation [50, 51, 52, 53].} \]

Towards LLM-assisted generation of separating concepts, we propose the following architecture, illustrated in the box below and summarised in Fig. 1 (left).
from sets of event logs, of a formal specification that captures the underlying process structure. In an LTL setting (we assume familiarity with its basic notions), such process mining task can be connected to the problem of finding a temporal specification, in the form of an LTL formula, that is capable of discerning a set of “positive” logs, i.e., examples of successful processes, from a set of “negative” ones, instantiating instead undesired dynamics. This problem has also received attention in the literature from a theoretical standpoint [37]. Similarly to the concept separability problem presented above, the task here is, given a set of propositional letters \( \Sigma \), a set \( P \subseteq (2^\Sigma)^\omega \) of positive traces, and a set \( N \subseteq (2^\Sigma)^\omega \) of negative traces, to determine a corresponding LTL process \( \Sigma \)-formula \( \varphi \) that: \( \sigma^+ \models \varphi \), for all \( \sigma^+ \in P \); and \( \sigma^- \not\models \varphi \), for all \( \sigma^- \in N \). An analogous problem would consider LTL on finite traces (often denoted by LTL\(^f\)), which is interpreted on finite sequences in \((2^\Sigma)^+\). Our LLM-driven architecture to address such a process formula generation is described in the following box, and illustrated in Fig. 1 (right).

The procedures sketched above are not guaranteed to terminate, as the LLM module might be incapable of finding successful candidates, and it is possible that no separating concept [35] or formula [56] exist in the formalisms. Further analysis on soundness, completeness, termination, and explainability issues, involving heuristic search techniques and reinforcement learning, or controlled loops with time-outs and generation of failed attempt explanations, is left as future work. For another recent approach integrating LLMs and declarative process mining, cf. [57].

![Figure 1](image_url)

**Figure 1:** Architecture of LLM-driven DL concept (left) and LTL formula (right) learning.

### 3. LLM-Driven Strategy Synthesis

With *strategy synthesis*, we refer to the problem of identifying a user strategy providing a sequence of actions to reach a goal, or of operations capable of satisfying a given specification, possibly in response to uncontrollable choices of other agents or environments. As such, it can encompass problems both in the fields of *planning*, as well as in *automatic programming*. 
For planning purposes, in a purely LTL setting, the so-called realisability and synthesis problems have attracted considerable attention [58, 59, 60], particularly in the finite trace case of LTL. Here, we slightly modify the standard setting [58, 61], and adopt the following definitions. Let \( \varphi \) be an LTL formula, with its proposition letters from \( \Sigma \) partitioned in sets of controllable (C) and Environment (E) ones. A strategy for \( \varphi \) is a function \( s : (2^E)^+ \rightarrow 2^C \) such that, for any finite sequence \( E = (E_0, \ldots, E_n) \in (2^E)^+ \) of Environment choices, it determines a Controller choice \( s(E) \in 2^C \). Moreover, let \( A \subseteq (2^E)^+ \) be a finite set of admissible infinite sequences of Environment choices. A strategy is winning if, for any admissible \( E \in A \), there exists \( k \in \mathbb{N} \) such that \( \text{react}(s, E)[0,k] = \varphi \), where \( \text{react}(s, E) = \{E_0 \cup s((E_0)), E_0 \cup s((E_0, E_1)), \ldots \} \) is the trace obtained by reacting to \( E \) according to \( s \), and \( \text{react}(s, E)[0,k] \) denotes its prefix from 0 to \( k \). An LTL formula \( \varphi \) is realisable with respect to \( (C, E, A) \) if there exists a winning strategy. The realisability problem asks whether \( \varphi \) is realisable with respect to \( (C, E, A) \), while the synthesis problem requires to provide such a winning strategy if it exists.

In the box below and in Fig. 2 we illustrate our architecture, aiming at synthesising LTL formulas via combined interactions between an LLM-based module and a model-checking tool.

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>((\varphi, C, E, A, N)), with (N) (possibly empty) set of traces not satisfying (\varphi).</td>
<td>Winning strategy for (\varphi) under (A), if it exists.</td>
</tr>
<tr>
<td><strong>Procedure</strong></td>
<td></td>
</tr>
<tr>
<td>• While no winning strategy for (\varphi) is found, repeat the following steps:</td>
<td></td>
</tr>
<tr>
<td>– prompt input ((\varphi, C, E, A, N)) to the LLM-based LTL formula synthesis module;</td>
<td></td>
</tr>
<tr>
<td>– ask the LLM module to candidate a strategy for (\varphi);</td>
<td></td>
</tr>
<tr>
<td>– check with a model-checking tool whether, for all (E \in A), (\text{react}(s, E) = \varphi):</td>
<td></td>
</tr>
<tr>
<td>• if a counterexample (E) is found, i.e., (\text{react}(s, E) \not= \varphi), assign (N \leftarrow N \cup {\text{react}(s, E)});</td>
<td></td>
</tr>
<tr>
<td>• otherwise, return (s) as winning strategy.</td>
<td></td>
</tr>
</tbody>
</table>

Observe that the set of admissible sequences of Environment choices is a restriction imposed to limit the search space in the formal verification module. Moreover, the LLM module should provide a finite presentation of the Controller strategy, by means e.g. of a finite-state transducer.

![Figure 2: Architecture of LLM-driven LTL strategy synthesis.](image)

### 4. Discussion and Future Work

We proposed architectures for the development of AI agents capable of performing complex knowledge acquisition and strategy synthesis tasks, combining the generative capabilities of LLMs with logic-based formalisms and techniques. As future work, we plan to both improve the definition and the understanding of the formal properties of the proposed architectures, as well as to develop dedicated tools based on state-of-the-art LLMs, comparing their performances over suitable benchmarks with other systems from the literature.
Acknowledgments

M. Robol and M. Roveri are partially supported by the project MUR PRIN 2020 - RIPER - Resilient AI-Based Self-Programming and Strategic Reasoning - CUP E63C2200400001. P. Giorgini, A. Mazzullo and M. Roveri are partially supported by the PNRR project FAIR - Future AI Research (PE00000013), under the NRRP MUR program funded by NextGenerationEU.

References


[28] Handbook of Temporal Reasoning in Artificial Intelligence, volume 1 of Foundations of...


[43] B. Settles, Active Learning, Synthesis Lectures on Artificial Intelligence and Machine


