Design Patterns for LLM-based Neuro-Symbolic Systems

Maaike de Boer[a], Quirine Smit[a,*], Michael van Bekkum [a], André Meyer-Vitali [b] and Thomas Schmid [c,d]
[a] *Dep. Data Science, TNO, The Hague, The Netherlands*
[b] *Deutsches Forschungszentrum für Künstliche Intelligenz GmbH (DFKI), Saarbrücken, Germany*
[c] *Medical Faculty, Martin Luther University Halle-Wittenberg, Halle (Saale), Germany*
[d] *Leipzig School of Computing and Communications, Lancaster University, Leipzig, Germany*

Abstract. Large Language Models (LLMs) have been a dominating trend in Artificial Intelligence (AI) in the past years. At the same time, neuro-symbolic systems employing LLMs have also received increasing interest due to their advantages over purely statistical generative models: they can make explicit use of expert knowledge and can be understood and inspected by humans thus providing explainability. However, with an increasing variety of approaches, it is currently difficult to compare the different ways in which designing, training, fine-tuning, and applying such approaches take place. In this work, we use and extend the modular design patterns for hybrid learning and reasoning systems and the Boxology language of van Bekkum et al. for this purpose. These patterns provide a general language to describe, compare, and understand the different architectures and methods used for LLM-based neuro-symbolic systems. The primary goal of this work is to support a better understanding of specific classes of such systems, namely LLM-based models that are used in conjunction with knowledge-based (symbolic) systems. In order to demonstrate the usefulness of this approach, we explore existing LLM-based neuro-symbolic architectures and approaches, as well as use cases for these design patterns.

Keywords: design patterns, neuro-symbolic AI, generative models, Large Language Models

1. Introduction

A major driving force for the origin and growth of the field of artificial intelligence (AI) has been the goal of "making computers solve really difficult problems" [57]. Today, many contemporary observers agree that AI has taken a leap in recent years [73] and has now reached a level of capacity and productivity that was unprecedented in previous decades [61]. Since 2010, AI systems have reached a close-to-human or even superior-to-human level in many computer vision tasks [40]. Since 2020, an increasing number of AI systems have been introduced that are able to successfully complete complex text generation tasks in natural language processing (NLP) [55], such as text summarization, translation, and question answering. More recently, the concept of generation has even been extended to multi-modal approaches involving, for example, text input and image output [6, 35, 68, 89, 106]. Many of these systems have demonstrated natural language processing capabilities at a level very close-to-human capabilities [113].

These developments are largely attributed to advances in deep learning techniques, in particular in the form of generative AI and Large Language Models (LLMs). A wealth of different LLM models have been and are being developed and published, both open-source and proprietary [10, 20, 42, 56]. The key technology most current

*Corresponding authors, both contributed equally: maaike.deboer@tno.nl, quirine.smit@tno.nl.

LLMs use is the transformer architecture. The original transformer architecture published by Vaswani et al. [87] proposed using two interacting models, an encoder and a decoder. These can be trained end-to-end (such as flan-T5 [13]). Alternatively, architectures have been proposed using encoder-only (BERT [19]) or decoder-only (GPT [8], BLOOMZ [59], PaLM [12]) models. As only a few LLMs based on other architectures have been proposed [4, 65], in this paper, we focus on transformer-based LLMs and consider encoder-only, decoder-only, and encoder-decoder systems to be possible types of LLMs.

These different categories of transformers provide different advantages and disadvantages, depending on the intended scenario. Encoder-only transformers, such as BERT [19], specialise in contextual encoding, often named base models. They use context to encode input sentences and represent them as machine-interpretable representations, such as vector representations. Decoder-only systems are complementary to the encoder-only paradigm, but structurally different [55]. A decoder-only system decodes the input data directly, without being transformed into a higher and more abstract representation to the desired representation (text, images, or otherwise). Examples of this are generative models from the GPT family [8]. Decoder-only architectures can be further divided into causal decoder architectures and prefix decoder architectures. Causal decoder architectures, such as GPT [8, 67] and BLOOMZ [59], use only unidirectional attention to the input sequence by using a specific mask. Prefix decoder architectures, such as PaLM [12], use bidirectional attention for tokens in the prefix while maintaining unidirectional attention to generate subsequent tokens.

Despite the many impressive achievements and capabilities of many LLMs, a wide variety of challenges remain for purely statistical LLMs [38]. This includes not only substantial costs for training [76] and inference [72], but also the infamous phenomenon of hallucination: situations where a trained LLM generates outputs presented as plausible or authoritative information that are factually incorrect, nonsensical, or unfaithful to the input or context [34]. In general, this phenomenon is attributed to the lack of training data for certain outputs, often indicating a lack of domain-specific knowledge [112]. While this may in part be corrected by retrieval-augmented generation (RAG) [25] and fine-tuning [107], they introduce novel challenges by themselves; fine-tuning, for example, may well raise significant additional training costs and may lead to catastrophic forgetting. A challenge in using RAG is that the performance is highly dependent on the quality and accuracy of the information retrieved. Moreover, both hallucinations and lack of domain-specific knowledge are often hard to identify or contextualize due to the inherent lack of explainability and interpretability of LLMs [63, 111]. It is therefore fair to state that current LLM-based systems and applications lack a sufficient degree of trustworthiness [32, 47], rendering them unusable where reliable output is essential (e.g., in scientific discovery [75]).

In response to these challenges, a variety of novel neuro-symbolic approaches to LLM-based AI systems have recently emerged [29, 94]. Due to the quantity and diversity of emerging generative techniques, it becomes increasingly challenging to keep track of the ever-growing variety of models with different LLM architectures and capabilities [14]. This becomes even more challenging with the growing diversity in combining LLMs with symbolic AI techniques using different strategies on different architectural levels and training stages [2, 18]. A practical solution to tackle the issue of analysing and understanding these approaches in a systematic way is to apply a high-level conceptual framework to discuss, compare, configure, and combine different models. Such a framework is provided by the Boxology, introduced by van Harmelen and ten Teije [86] in 2019. The Boxology was extended in 2021 [84] by providing a taxonomically organised vocabulary to describe both processes and data structures used in hybrid systems. The Boxology represents a flexible and widely applicable framework for representing AI design patterns.

In this paper, which is an extension of our previous work on LLM-based neuro-symbolic systems [17], we propose to use and extend Boxology to gain insight into a variety of LLMs, specifically on LLMs used in a neuro-symbolic approach. To this end, this paper provides two contributions: firstly, we propose novel design patterns as an extension of the current Boxology to promote transparency and trustworthiness in system design, by providing interpretable, high-level component descriptions of LLM-based neuro-symbolic systems. Our modular approach supports new architectures and engineering approaches to LLM-based systems. Secondly, we test validity and usefulness of the Boxology and our extensions in this field on example architectures and applications, such as ChatGPT, KnowGL, GENOME and Logic-LM.

The remainder of the paper is organised as follows. In the next section, we give a more detailed overview of the related work regarding LLMs and LLM-based neuro-symbolic systems. In the third section, we propose to extend

the Boxology by three novel basic patterns in order to be able to handle LLMs, and in the fourth section we explain several compositional design patterns in this field. In section 5, we dive into specific applications and tasks in which LLMs, specifically in neuro-symbolic systems, are used. We conclude with a discussion (section 6) and a conclusion summarising our key findings and outlining future work in section 7.

2. Related Work

2.1. Neuro-Symbolic Systems and Design Patterns

For a long time, the field of AI developers had been shaped by an opposition between "imitators of the mind" favouring symbolic AI approaches and "imitators of the brain", favoring statistical approaches, such as artificial neural networks [16]. Today, however, combining the complementary strengths and weaknesses of symbolic and statistical approaches is considered crucial for the creation of reliable, trustworthy, and effective AI systems [51]. After a first wave of symbolic approaches and a second wave of statistical approaches, the combination of symbolic and statistical approaches is therefore anticipated as the latest, third wave of AI [26, 85]. Starting in the early 1990s, neuro-symbolic AI has been emerging with a large variety of diverse approaches. This characteristic has been recognised early with descriptive classification approaches [28, 53] and workshops that showcase and celebrate the diversity of neuro-symbolic systems [81]. A basic yet widely recognized and applicable taxonomy published by McGarry et al. in 1999 differentiates three main classes of hybrid neural systems [52]: unified hybrid systems, translational hybrid systems and modular hybrid systems.

Unified and translational hybrid systems as defined by McGarry have been a popular area of study within the neuro-symbolic AI community during the last two decades [5, 105]. At the same time, modular hybrid systems have also gained wider interest due to its relevance for designing modern AI systems under industrial conditions [74]. In general, the design paradigm of modularization is characterised and motivated by the possibility to reuse and/or reorganise modules. Identifying and characterising design patterns for modular approaches of neuro-symbolic has led to the development of the so-called Boxology framework described below [84, 86]. This framework has been used and extended in different ways, such as the formalisation of the notions from the Boxology and implementation in the heterogeneous tool set [58], the extension of the Boxology for (teams of) actors [54], the characterisation of emerging data-driven knowledge engineering trends [71], and the systematic study of nearly 500 papers published in the past decade in the area of Semantic Web Machine Learning [7]. While the original Boxology framework focused on architectural aspects of hybrid learning and reasoning, it has been pointed out the necessity of including representations of human involvement into such patterns [54, 96]. Despite the usefulness and manifold advantages of the above-mentioned traditional taxonomies and studies, however, it must be stated that none of these are able to foresee and fully capture the characteristics of the latest generation of neural systems [17], in particular the broad capabilities and the required extraordinary architectural complexity of LLMs.

A recent approach to describe modular neuro-symbolic AI systems is the ontological visual framework termed EASY-AI, which uses semantically enhanced symbols to represent the components and architectures of the AI system [21]. EASY-AI aims to provide a standardised symbolic language for conveying the structure, purpose, and characteristics of AI systems. The approach presents the logical formalisms underpinning this visual framework, with the objective of enhancing the comprehensibility and understandability of AI system behaviours. Recently, this framework has also been provided with an initial implementation named SNOOP-AI [22]. This framework and implementation could be used in the implementation of the design patterns, as it can provide a formal conceptual foundation for the design patterns that allows formal reasoning over (compositions of) its elements. To the best of our knowledge, specific LLM-based use cases have not been tested using formalisation and implementation yet.

2.2. LLM-based Neuro-Symbolic Systems

While older frameworks for neuro-symbolic AI have not been designed to reflect the latest generation of complex large-scale neural components in the first place, some alternative approaches for combining LLMs and knowledge-based components have been designed from scratch. Colon-Hernandéz et al. were among the first to to do so by

identifying three different categories of so-called knowledge injections [15]: approaches to modify the architecture of LLM by adding additional layers that integrate knowledge with contextual representations or by modifying existing layers (termed architectural injections) are distinguished from approaches aiming to modify either the structure of the input or the data selected to be fed into the LLM (referred to as input injection) and approaches to change either the output structure or the losses that were used in the base model in some way to incorporate knowledge (termed output injection). While providing new insights and perspectives for LLM-based neuro-symbolic systems, however, this injection-oriented approach is centered largely around modifications of LLMs, leaving out the modular perspective of integrating equally relevant neural and symbolic components.

In light of the popularity and success of knowledge graphs (KG) in recent years [33], it is not surprising that this symbolic technique is a prime candidate for many researchers thriving the enhance LLMs. To this end, Agrawal et al. have suggested to distinguish between LLM-based systems with knowledge-aware inference (KG-augmented retrieval, KG-augmented reasoning, or KG-controlled generation), knowledge-aware training (pre-training or fine-tuning), and knowledge-aware validation [1]. While their taxonomy represents a rather empirical high-level categorization, an alternative approach by Pan et al. [63] provided a more differentiated approach where a distinction is made between KG-enhanced LLMs, LLM-augmented KGs and synergised LLMs + KGs. For KG-enhanced LLMs, two primary approaches have been explored: incorporation during the pre-training stage to facilitate knowledge acquisition and utilisation during the inference stage to improve access to domain-specific information. Additionally, KGs have been employed post-hoc to augment the interpretability of LLMs, elucidating both factual content and reasoning processes. In order to augment KGs, LLMs have been employed as text encoders to enrich KG representations and extract relations and entities from the original corpora. Recent studies have focused on designing KG prompts that effectively convert structural KGs into LLM-comprehensible formats, allowing direct application of LLMs to KG-related tasks such as completion and reasoning. Moreover, the authors have proposed considering the effects and concepts of synergised LLM + KG with respect to four layers: 1) Data, 2) Synergised Model, 3) Technique, and 4) Application. We will loosely use the categorisation of this paper in our exploration of different LLM-based neuro-symbolic systems.

3. Extending the Boxology framework with novel elements

We base our paper on the previous work of van Bekkum and colleagues [84], in which a taxonomically organised vocabulary is provided to describe both processes and data structures used in neuro-symbolic systems. The highest level of this taxonomy contains instances, models, processes, and actors, which may be described as follows:

Instances: The two main classes of instances are data and symbols. *Symbols* are defined as having a designation to an object, class, or relation in the world that can be atomic or complex, and when a new symbol is created from another symbol and a system of operations, it should have a designation. Examples of symbols are labels (short descriptions), relations (connections between data items, such as triples), and traces (records of data and events). *Data* is defined as not symbolic. Examples are numbers, texts, tensors, or streams.

Models: Models are descriptions of entities and their relationships, which can be statistical or semantic. *Statistical* models represent dependencies between statistical variables, such as LLMs or Bayesian Networks. *Semantic* models specify concepts, attributes, and relationships to represent the implicit meaning of symbols, such as ontologies, taxonomies, knowledge graphs, or rule bases.

Processes: Processes are operations on instances and models. Three types of processes are defined: generation, transformation, and inference. *Generation* can be performed using, for example, the training of a model or by knowledge engineering. *Transformation* is the transformation of data, for example, from a knowledge graph to vector space. *Inference* can be inductive or deductive, in which induction generalizes instances and deduction reaches conclusions on specific instances, such as classification.

Actors: Actors can be humans, (software) agents, or robots (physically embedded agents). [54] extended the original paper with a definition of teams of actors in the Boxology.

In addition to vocabulary, visual language is defined in [84], as an extension of [86]. The visual language consists of rectangular boxes (instances), hexagonal boxes (models), ovals (processes) and triangles (actors), and untyped arrows between them. Within the boxes the concept will be noted by each level in the vocabulary using colon-separation from most generic to most-specific, for example a neural network will be `model:stat:NN`. In the figures we use the notation of most generic and more specific only to improve readability, so `model:NN`.

3.1. Introducing a new elementary pattern

The classic Boxology framework [84] is based on eight elementary patterns (cf. Figure 1). The elementary patterns 1a-1d are elementary to generate a model, in particular for both statistical as well as semantical models. For instance, Pattern 1a shows how to train a statistical model (such as a neural network) using data (such as text or images). Pattern 1b shows how to create a semantic model (such as a knowledge graph) using symbols (such as triples). The elementary patterns 2a-2d are patterns describing how to use a model. Pattern 2a, for example, shows a statistical model (such as a neural network) being used to deduce symbols, i.e. for a classification task. Pattern 2b depicts the application of a semantic model, for example when using it for reasoning.

When seeking to apply any of the eight elementary patterns of the classic Boxology framework to represent LLMs, however, it becomes apparent that such generative AI approaches are not adequately represented in these elementary patterns. While the main characteristic of generative AI systems is the ability to output new data for given input data, in the existing elementary patterns only symbols or models can be inferred. Therefore, we propose to extend the eight elementary patterns (Figure 1, 1a-1d and 2a-d) by introducing a new additional elementary pattern 2e (Figure 1). In contrast to the existing elementary patterns, this novel pattern allows to represent LLM-based neuro-symbolical systems since this pattern represents a model that can infer new data from data. This new data can be an image, video, or text, depending on the type of model. For example, with GPT [8], LLaMa [20] and similar generative text models, new text is generated based on given input text.

While we focus on textual models in this paper, it is worth mentioning that the pattern proposed in this section abstracts from the specifics of the type of data. The new elementary pattern 2e is thus transferable to other generative models and data types and applies also for example to image generation models [6, 35, 68, 89, 106], which can generate image data from text data. Specifically this would mean that the type of input data would be `data:text` and the type of output data would be `data:image`.

3.2. Introducing a new compositional pattern

One of the key characteristics of the Boxology framework is its modularity, allowing the combination and reuse of elementary patterns in compositional patterns. Van Bekkum et al. describe, for example, the two compositional patterns 3a and 3b depicted in Figure 2 [84]. For the compositional pattern 3a, the two elementary patterns 1a and 2a (Figure 1) are combined. The resulting compositional pattern describes a basic structure for a (statistical) machine learning model depicting the training (creation of the model) and testing or application phase (application of the model on new data). Similarly, the compositional pattern 3b is created from combining two elementary patterns, allowing to depict a basic structure for a semantic model.

When seeking to use any of the two compositional patterns 3a and 3b as a combined representation for training and application of an LLM, however, it becomes apparent that – just as with elementary patterns for LLM-based training and inference – the compositional patterns of the classic Boxology framework are not well-suited for this. Thus, we introduce the new compositional pattern 3c (Figure 2) for the combined training and application of an LLM. This additional pattern is built from a combination of the two elementary patterns 1a and 2e (Figure 1). Similar to the novel elementary pattern 2e, this novel compositional is not limited to text-based data only, but can also be applied to any type of data, including image data, audio data, and multimodal data.

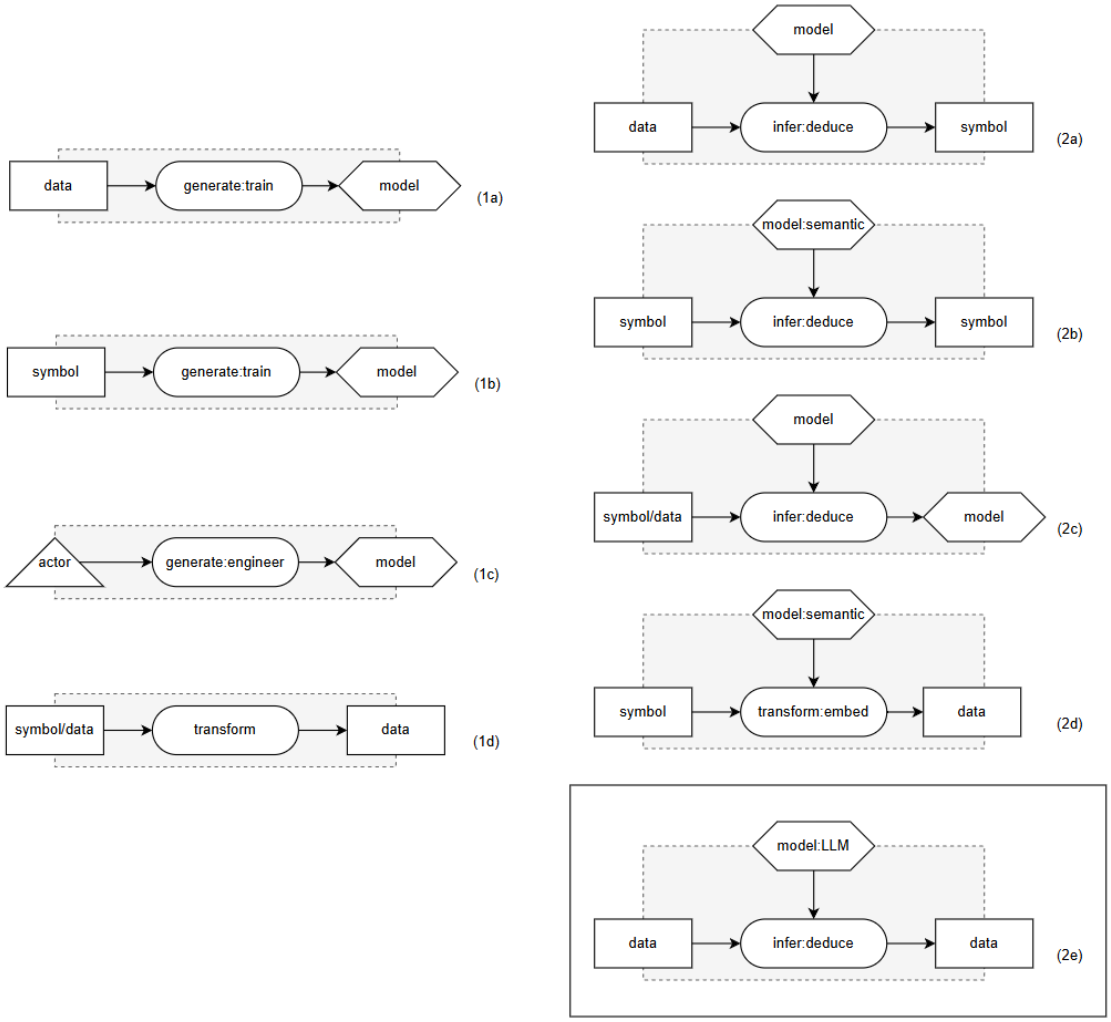

Fig. 1. All elementary design patterns, including novel addition 2e. Patterns 1a to 1c allow for model generation, 1d for transforming data and patterns 2a to 2e allow for model use.

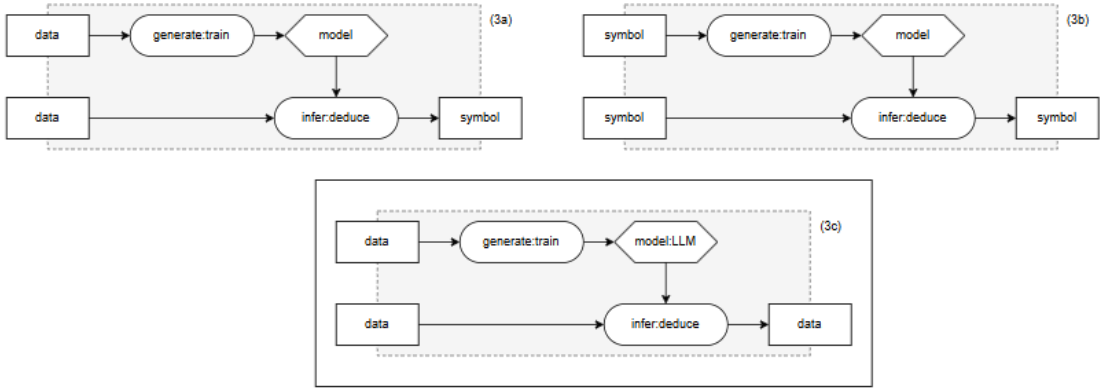

Fig. 2. Compositional design patterns, including novel addition 3c made by combining elementary pattern 1a and 2e. Patterns 3a and 3b visualize the patterns for full learning and prediction tasks from data/symbols.

4. LLM-based Neuro-Symbolic Design Patterns

Current LLM-based neuro-symbolic systems often either use an LLM followed by a semantic model, or a semantic model followed by an LLM, or a combination of two models in parallel of which the output is fused. In this section, we propose compositional design patterns for these different types of system. We loosely follow the categorisation of Pan et al. [63]. We divide the section into training and application phases, as the patterns for these phases are distinct.

4.1. LLM-based Neuro-Symbolic Design Patterns in Training

Generative neuro-symbolic systems can use semantic models in the training of an LLM or use an LLM to create a semantic model, or can be used in synergy to create a model. In the following subsections, we will describe the different patterns in more detail.

4.1.1. KG-enhanced LLMs

KGs can be used to enhance LLMs in training, for example by influencing the training data. An example of this is shown in the design pattern in Figure 3. Here, a KG is used to infer symbols (pattern 2b). These symbols are then changed into data (pattern 1d). This data is then used to train the LLM (pattern 1a). This depiction can be used to represent [44, 69, 77, 98], for example when a KG is used when masking the data to improve the training of the LLM. For example, in GLM [77] the masking probability is higher for concepts which are close together in the KG. On the other hand, SKEP [83] uses a KG to identify words of high sentiment and gives them a higher masking probability.

4.1.2. LLM-augmented KGs

LLMs can be used to enhance KGs as KGs might be incomplete and textual information is not integrated in the embedding itself. It can be represented in Boxology as presented in Figure 4. Similar to the KG-enhanced LLMs in training, new data is infered by an LLM (pattern 2e). These are then transformed to symbols (1d) and used to create or add on to a KG model (1a). For example, Nayyeri et al. [60] generate representations on different levels such as sentence and document using LLMs and Huang et al. [31] create multi-modal embeddings. Models following this structure are often used for tasks such as LLM-augmented KG completion and construction, including Named Entity Recognition, Coreference Resolution, and Relation Extraction. For example, KG-BERT, MLT-KGC, and PKGC use LLMs for the completion of a knowledge graph [41, 50, 101]. They use the LLM output to predict the relation between new entities and existing ones. Yan et al. [99] uses LLMs to aid in Named Entity Recognition, [9, 37] for Coreference Resolution, and [64, 78] for Relation Extraction.

4.1.3. Synergised LLMs and KGs

One of the ways in which LLMs and KGs are synergised in training is using an LLM for joint text and KG embedding or representation. Figure 5 shows the Boxology representation of these types of systems. The symbolic triples are transformed into text (pattern 1d), which is then combined with other text to integrate both the graph structure and the textual information into the embedding simultaneously and trained to create a model (pattern 1a). For example, kNN-KGE sees entities as special tokens and incorporates them into sentences as input for the LLM [90]. LMKE has a similar system structure but applies a different learning method to improve the learnt embeddings [92]. LambdaKG improves the representation of the graph structure by including neighbouring entities in the input sentence [97]. KEPLER, JointGT and DRAGON use a unified model for the knowledge embedding and pre-trained language representation [39, 91, 103]. They have pre-training tasks to come to a joint knowledge embedding and language modelling optimisation. ERNIE proposes a dual encoder system, consisting of a textual encoder that is fused with the knowledge graph encoder [109]. BERT-MK has a similar dual encoder, but adds additional information from neighbouring entities to the knowledge graph [27]. Coke-BERT further improves on this idea by adding a module to filter out irrelevant neighbouring entities [79]. JAKET fuses the entity representation in the middle layers of the LLM [104].

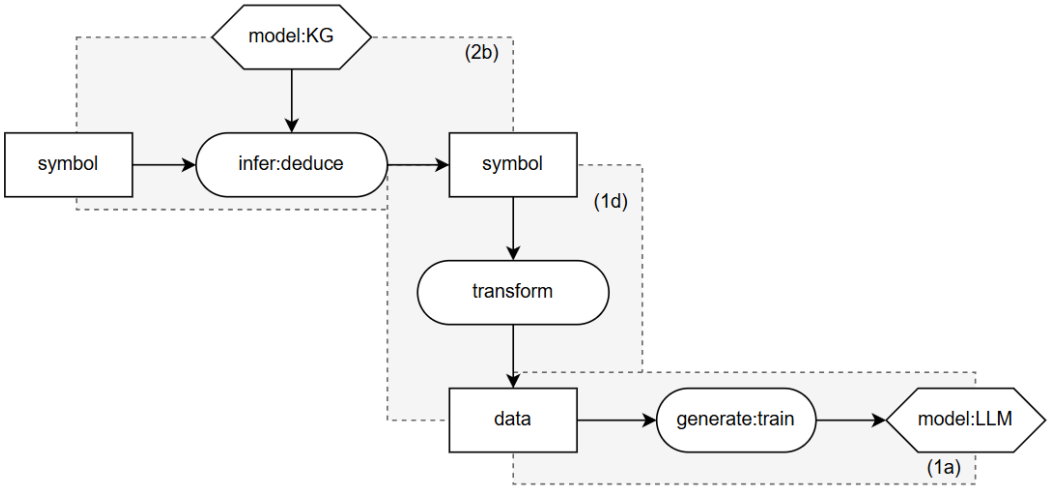

Fig. 3. KG-enhanced LLMs in training.

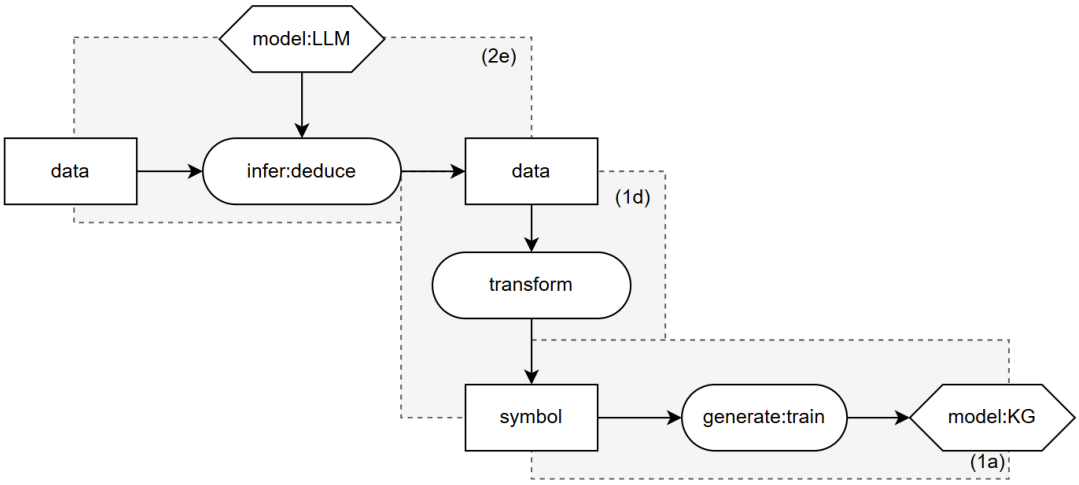

Fig. 4. LLM-augmented KGs in training

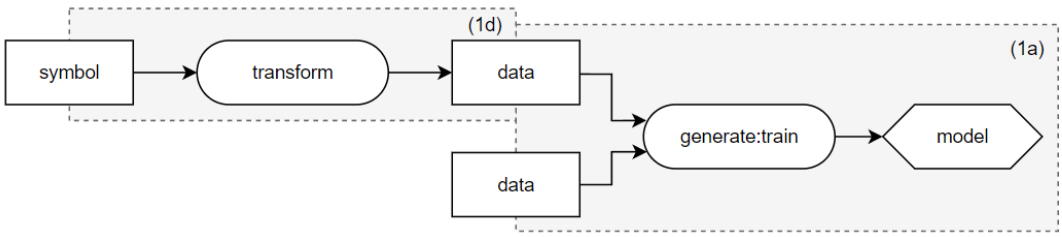

Fig. 5. Synergised LLMs and KGs in training

4.2. LLM-based Neuro-Symbolic Design Patterns during Inference

Neuro-symbolic systems often combine KGs and LLMs during inference, after training. In this way, the system is more robust to new situations. Many of the LLM-based neuro-symbolic systems follow one of the pre-defined patterns. This section will highlight three depictions of the LLM-based NeSy systems during inference.

4.2.1. KG-enhanced LLMs

KGs can be used to enhance LLMs by utilizing the knowledge in KGs. One way to do this is represented in Figure 6. It shows how a KG is used to infer symbols (pattern 2b). These symbols are then transformed to data (pattern 1d) which is used by the LLM to generate new data (pattern 2e). This can be useful for example to align the input data with the knowledge or augment it by adding relevant facts for the LLM to improve the output. In contrast to KG injection during training (see subsubsection 4.1.1), the results of pattern 2b and 1d are now input to the infer process instead of the train process. This means that the knowledge is up to date at the time of inference, rather than at the time of training, which may happen a long time before deployment.

This pattern describes systems that transform the input data by aligning them with the knowledge of the KG before they are fed into the deduction process with an LLM model. This can be done in a prompt engineering process using KGs [45, 49, 88, 95] or retrieval-augmented knowledge methods such as RAG [43]. KagNet first encodes the input KG and then augments it with textual representation [46].

4.2.2. LLM-Augmented KGs

LLMs can be used to augment KGs to improve information deductions (Figure 7). An LLM is used to infer data (pattern 2e). Then the data is transformed to symbols (pattern 1d) for the KG to reason over (pattern 2b). As with KG-enhanced LLMs during inference, the difference between training and inference for LLM-augmented KGs is that the first pattern is input to infer process rather than to train process of the KG.

One example is using LLMs for KG embedding. Pretrain-KGE uses an LLM to encode the text of the parts of the triples and uses that encoding as a starting point for the KG encoding [110]. Moreover, in answering questions with LLM-augmented KG, LLMs are used to bridge the gap between natural language questions and the retrieval of KG answers [30, 48]. In addition, LLMs can be used for the generation of text from a knowledge graph, where LLMs are used to generate natural language that describes facts from KGs [23, 80, 93]. MHGRN uses the LLM representation of the text to guide the reasoning process in the KGs [24].

4.2.3. Synergised LLMs and KGs

LLMs and KGs can be combined to work in synergy, also in the application phase. Figure 8 shows how this can be applied, specifically in the case of synergised reasoning. Here, the model is fed both symbols and data, both in the training and in the application phase.

Examples of such methods are JointLK [82] and GreaseLM [108]. They include interactions between the tokens in the textual input and the entities in the graph in the model's layers. QA-GNN [102] represents the LLM information as a special node in the KG for reasoning.

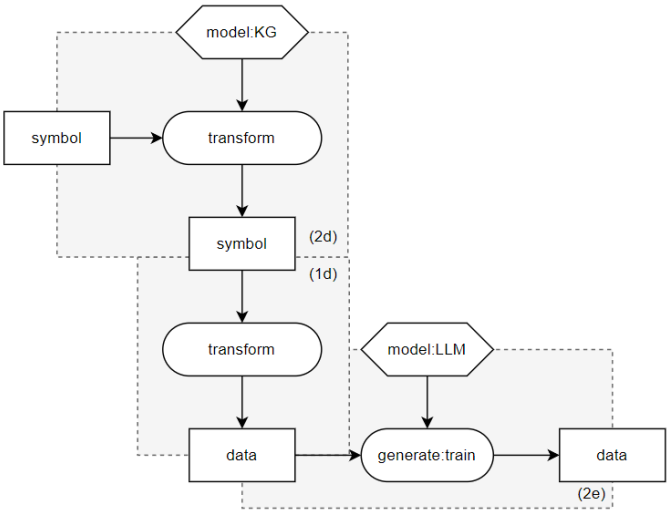

Fig. 6. KG-enhanced LLMs during inference

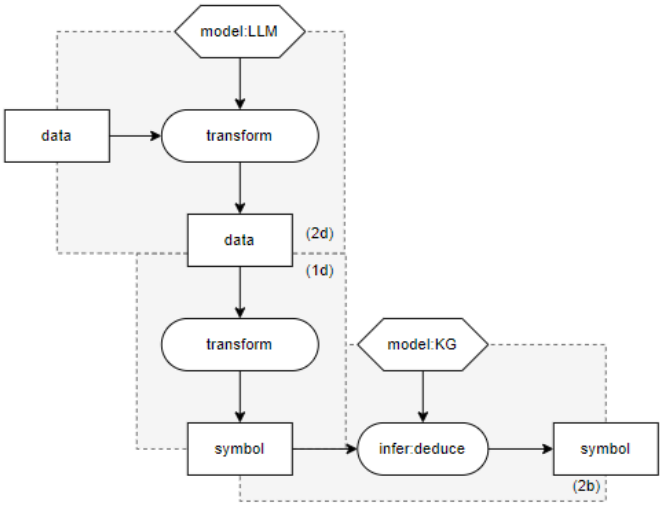

Fig. 7. LLM-augmented KGs during inference

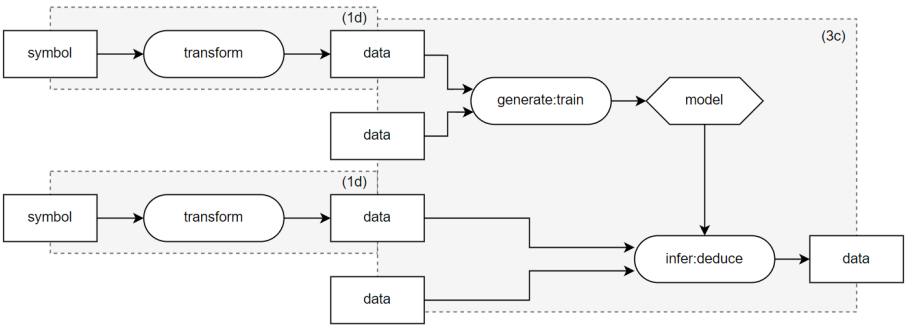

Fig. 8. Synergised LLMs and KGs for reasoning

5. Use Cases

In this section, we describe and explore several papers that propose an LLM-based neuro-symbolic system. The selected papers are chosen, as they represent a diverse set of possibilities to use an LLM in a system pipeline (on the input side of the system, somewhere in the inner part, or on the output side) as well as act as a fluent language interface or a formal language interface on the input or output side.

5.1. Retrieval-Augmented Generation (RAG)

Retrieval-Augmented Generation (RAG) is a method that expands an LLM with external knowledge [43]. A RAG system has two main components, a retriever and a generator. Figure 9 shows the Boxology representation of a RAG system, where the retriever is the model in pattern 2a and 1d and the generator is the LLM in pattern 2e. Firstly, the retriever selects relevant documents based on the posed question (pattern 2a), through classification or with help of a Knowledge Graph. Secondly, the question and retrieved documents are transformed (1d) to be presented to an LLM in a prompt (pattern 2e). Thirdly, the LLM generates an answer to the question based on the information in the selected documents. The LLM can also present the source of the information, making it more trustworthy and reliable.

In KD-CoT, KSL and Think-on-graph, facts are retrieved from a KG together with the reasoning, and an LLM generates a natural language answer to be presented to the user [23, 80, 93]. RAG is a prime example of the KG-enhanced LLM pattern, as presented in subsubsection 4.2.1.

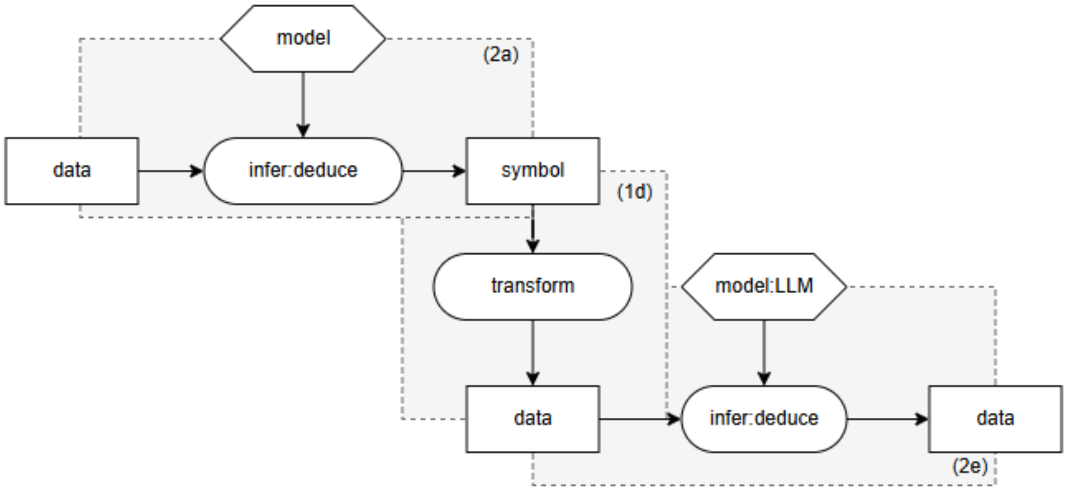

Fig. 9. Use of Retrieval-Augmented Generation

5.2. KnowGL

The KnowGL parser is a system developed by IBM Research for converting data into symbols. More specifically, it can be used to automatically extract knowledge graphs from collections of text documents [70]. KnowGL employs an LLM to extract semantic triples from each sentence, which are then enriched with semantic annotations. Figure 10 shows the Boxology representation of the KnowGL parser. Pattern 2e represents the BART-large model receiving a sentence and inducing a list ‘subject, relation, object’. In the next step, represented by pattern 2b, a ranked list is created of distinct facts and their scores. In the final step, the generated facts are linked to Wikidata. This is done using a mapping of labels to Wikidata IDs (pattern 2b). In the case that the LLM has created a new entity, type, or relation label that are not in Wikidata it returns ‘null’.

The architecture of the KnowGL parser displays a variation of the LLM-augmented KG in inference pattern, in subsubsection 4.2.2.

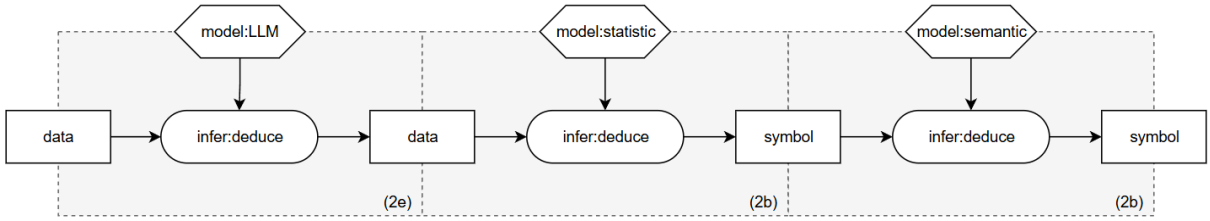

Fig. 10. Boxology representation of KnowGL [70]

5.3. *KnowBERT*

Although knowledge is mostly injected into statistical generative models during the input or during the output stage, approaches to inject knowledge inside the model have also been proposed. A prominent example is KnowBERT, a modified variant of BERT [66]. It stands out for its fusion of contextual and graph representations, attention-enhanced entity-spanned knowledge infusion, and flexibility in injecting multiple Knowledge Graphs at various model levels. KnowBERT embeds multiple knowledge bases (WordNet and a subset of Wikipedia) into LLMs to enhance their representations with structured, human-curated knowledge. By integrating the Knowledge Attention and Recontextualisation layers [3], graph entity embeddings are used that are processed through an attention mechanism to enhance entity span embeddings. This happens in later layers of the model to stabilise training, but can also potentially be used to inject knowledge at earlier stages [15]. The Boxology pattern for KnowBERT is shown in Figure 11 and is the same as the pattern presented in subsubsection 4.1.1 (KG-enhanced LLMs in training). Pattern 2b represents the incorporation of knowledge bases (KB) into a pre-trained BERT model, using an integrated entity linker, as shown with pattern 1d. Finally, the Knowledge Attention and Recontextualization component is the heart of KnowBert, which is represented as pattern 1a.

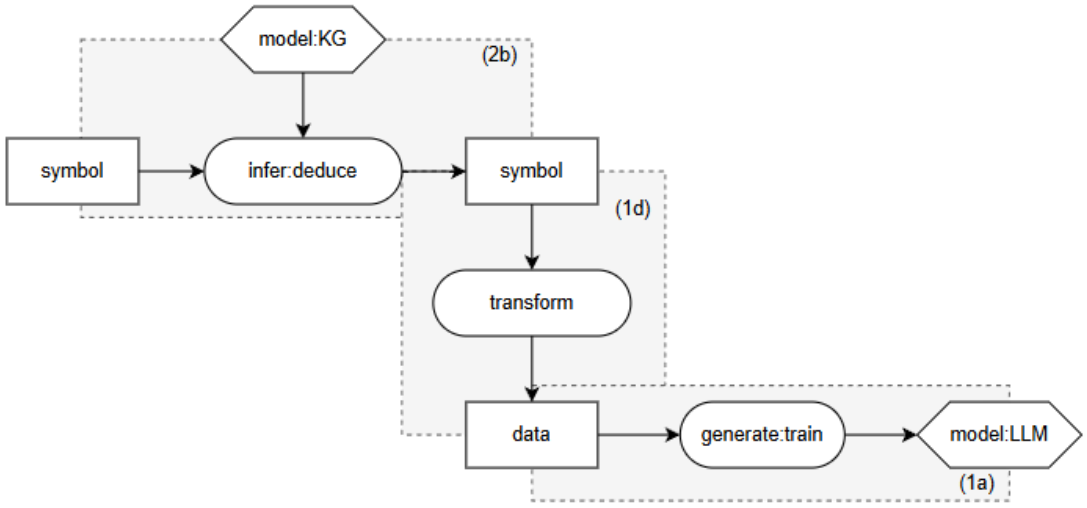

Fig. 11. Boxology representation of KnowBERT [66]

5.4. *Mathematical Conjecturing and LLMs*

Theory explanation or automated conjecturing is the process of inventing new conjectures about a set of functions. The system [36] has two principal components: 1) the system assigns the generative task of discovering mathematical conjectures to an LLM, 2) the results are checked using a symbolic theorem prover or counterexample finder.

The LLM is first trained on data from a formal language (pattern 3c). The system is then prompted with a formal theory (e.g. a sort function), and has the LLM generate lemmas from the theory as output data. These generated lemmas are transformed from data to symbols (pattern 1d) and are subsequently used by a semantic model(s) prover (pattern 2b). The Boxology representation is depicted in Figure 12.

The approach taken in Yang et al. [100] is also captured by this representation. The system proposed first uses an LLM component that has been trained on Prolog to generate Prolog code as output (pattern 3c). The output is then transformed to symbols (1d), a symbolic inference engine then produces answers and reasoning traces by executing the code mentioned above (pattern 2b).

Both of these examples show a generalisation of the Boxology pattern in subsubsection 4.2.2 (LLM-augmented KGs during inference), where the KG is replaced by a different semantic model.

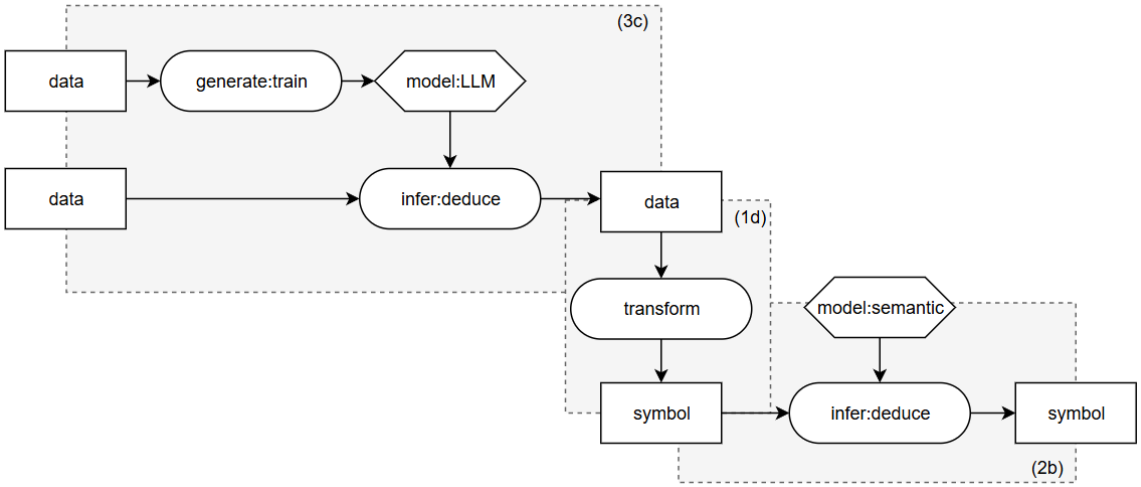

Fig. 12. Boxology representation for using LLMs for discovery of mathematical conjectures [36]

5.5. GENOME

Generative Neuro-Symbolic Visual Reasoning by Growing and Reusing Modules (GENOME) focuses on the task of generative software module learning [11]. Its architecture is based on one LLM generating signatures (input/output) for these software modules and reasoning steps, while another LLM subsequently creates the targeted software module based on those; as usually, both LLMs have been created by means of pre-training. Finally, GENOME employs a deductive reasoner to evaluate the LLM-generated module on a test case.

Figure 13 shows the Boxology representation of GENOME. The system consists of three stages, module initialisation, module generation and module execution, represented by two compositional and one elementary pattern. First, an LLM assesses a visual-language question and outputs new module signatures and operation steps as a response to the query (pattern 3c), if current modules from a library cannot provide an adequate response. In the next step, the LLM generates a module (software code) based on the signature/test case (pattern 3c, 2nd component). Finally, the module is executed by passing it a visual query (pattern 2a).

This use case is an extension of the LLM-augmented KGs during inference as described in subsubsection 4.2.2, deploying two LLMs in sequence.

5.6. Logic-LM

The approach taken by Logic-LM [62] integrates LLMs as a natural language interface with symbolic solvers to improve logical problem-solving. The logical problem can e.g. be stated as logical programming, first-order logic or a constraint satisfaction problem. This approach is depicted in Boxology notation in Figure 14. Here the system uses

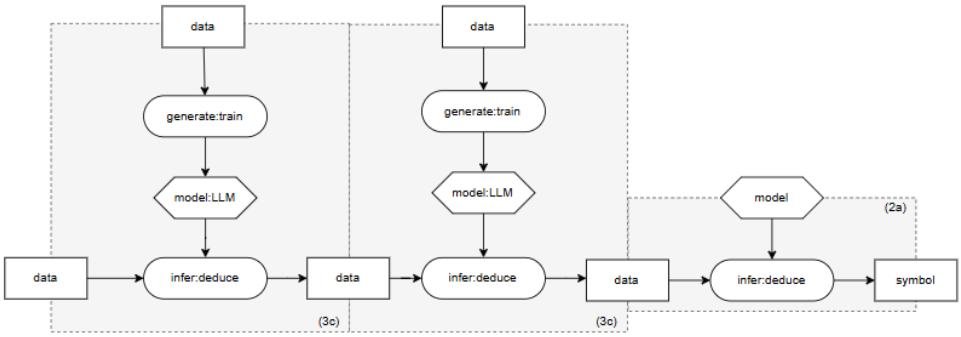

Fig. 13. Boxology representation of GENOME [11]

LLMs that are trained in the specific logic language (pattern 3c) to translate a problem stated in natural language into a symbolic formulation (patterns 3c and 1d). In the next step, a symbolic reasoner module performs logical inference on the formulated problem and transforms the symbolic results into data (patterns 2b and 1d), using a semantic model and a transformation of logical symbols to a prompt. Finally, an LLM receives the results as an input prompt and outputs a solution in natural language (pattern 3c). The LLM thus functions as a fluent language interface to and from a symbolic reasoner component. The main reasoning is performed by a logic engine (symbolic reasoner), in order to guarantee correct and verifiable results.

This representation is a combination of the patterns presented in LLM-augmented KGs during inference, subsubsection 4.2.2, and KG-enhanced LLMs during inference, subsubsection 4.2.1.

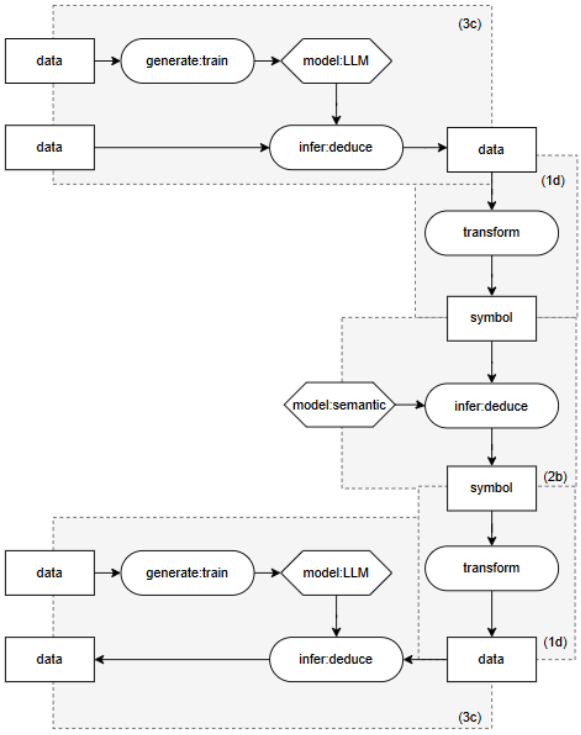

Fig. 14. Boxology representation of Logic-LM [62]

6. Discussion

Advancements. A key contribution of this work is the introduction of novel design patterns. In particular, we introduce a novel elementary pattern and a novel compositional pattern to complement the existing Boxology patterns. The novel patterns proposed in this paper provide a more fine-grained description of the model element, introducing the generative model. The novel elementary pattern allows, for the first time, to use the Boxology to represent data in - data out processing, which is the key concept underlaying generative AI techniques. Although we apply it exclusively to LLMs, we have argued that its level of abstraction lends itself to representing other types of generative models, such as generative image models. As a natural derivate of this, the newly introduced compositional pattern employed this elementary pattern to integrate training and inference for generative models. Providing three principled integrations of LLMs and KGs, we illustrate how the composition of elementary patterns can be used to describe LLMs, and we explore several categories as well as specific approaches in use cases, such as KnowGL, GENOME and Logic-LM.

Conceptualization. Whereas our extension demonstrates its applicability in describing key architectural features of LLM-based neuro-symbolic systems, it raises interesting questions about the nature of other Boxology elements directly related to the new element representing generative models (`model:generative`). The classical opposition of data and symbols, for example, has been a long-standing premise of the Boxology framework. Considering the input and output related to generative models, however, one may wonder whether the output is actually of the same nature. While no straightforward answer seems to be at hand, the framework might benefit from revisiting the underlaying nature of data versus symbols. At the same time, we notice that the deductive inference element perhaps generalises from the finer details of the inference capabilities of a generative model to a certain extent. The probabilistic nature of these models may well be captured by a finer distinction in the Boxology inference concept, which can perhaps also cover new classes of inference that have so far not been covered.

Ambiguity. In the process of applying basic design patterns to specific use cases, naturally questions arise about which pattern combinations are allowed and which are not. However, the truth is that Boxology will not in all real-world LLM-based neuro-symbolic systems lead to one unique and unambiguous representation. In practical classroom settings, for example, we have experienced the updated Boxology framework to yield multiple options for describing the same system, depending on the individual student or engineer. Much like some of the notations of UML have proven to lend itself to different designs of the same system based on different points of view, this phenomenon is likewise inherent to the Boxology framework. In order to reduce some of the ambiguity, a more formal description along with guidelines for interpretation and use of the current set of Boxology elements seems warranted.

7. Conclusion and Future Work

Despite many open questions and challenges towards generative AI techniques, LLMs are widely used in a variety of applications nowadays. To this end, combining data-driven approaches with knowledge-based techniques is a promising development to address these challenges. In this paper, we propose new design patterns for modular LLM-based neuro-symbolic systems to be included in the design pattern approach for neuro-symbolic systems as proposed by van Bekkum et al. [84]. Thereby, we are filling a gap in the classic Boxology framework with respect to the recent rise of generative AI techniques. To this end, this paper proposes an extension in terms of concepts and patterns to the set of Boxology elements and patterns as described in earlier work on Boxology [17, 84, 86]. Specifically, our goal is to make the Boxology framework compatible with the concepts underlaying LLMs and LLM-based neuro-symbolic systems. Given the fact that many existing, as well as many potential real-world applications are based on this disruptive AI paradigm, the extension provided within this paper can be considered a substantial conceptual update allowing to maintain the Boxology framework's relevance in the years to come.

In future work, we plan to further explore the usability and benefits of the updated Boxology and its design patterns in domains adjacent to LLM-based neuro-symbolic systems, such as generative AI systems without symbolic AI attached and multi-modal generative AI systems. Moreover, we anticipate the need to further extend and deepen the Boxology framework itself. Temporal or recurring/iterative aspects, for example, have not yet been taken into

account and can currently not be visualised adequately in this respect. Our current investigation has also shown that the current concept naming, concept labelling and some of the formalisation of the Boxology could benefit from critical review and in-depth revisiting. The importance of representing or modelling datasets, for example, may be taken into account in future specifications of particular subtypes of instances and models. Finally, we consider a future use of graphical tools for the Boxology beneficial. In software development, this approach has proven both efficient and effective and is well known, for example, from the Unified Modelling Language (UML) and visual programming tools, such as LabView or Scratch. And while our current work is mostly concerned with graphical representations of design patterns for system design and documentation, the promise of templates, low-code, or no-code development, seems also an appealing field of research for the future.

Acknowledgements

We thank the TNO project GRAIL for their financial support and Frank van Harmelen and Annette ten Teije for their feedback. We also thank Daan Di Scala for his contribution to the KnowGL pattern.

References

[1] G. Agrawal, T. Kumarage, Z. Alghamdi and H. Liu, Can knowledge graphs reduce hallucinations in llms?: A survey, *arXiv preprint arXiv:2311.07914* (2023).

[2] E. Amador-Domínguez, E. Serrano and D. Manrique, Neurosymbolic system profiling: A template-based approach, *Knowledge-Based Systems* **287** (2024), 111441. doi:https://doi.org/10.1016/j.knosys.2024.111441. https://www.sciencedirect.com/science/article/pii/S0950705124000765.

[3] I. Balažević, C. Allen and T.M. Hospedales, Tucker: Tensor factorization for knowledge graph completion, *arXiv:1901.09590* (2019).

[4] M. Beck, K. Pöppel, M. Spanring, A. Auer, O. Prudnikova, M. Kopp, G. Klambauer, J. Brandstetter and S. Hochreiter, xLSTM: Extended Long Short-Term Memory, *arXiv preprint arXiv:2405.04517* (2024).

[5] T.R. Besold, A. d'Avila Garcez, S. Bader, H. Bowman, P. Domingos, P. Hitzler, K.-U. Kühnberger, L.C. Lamb, P.M.V. Lima, L. de Penning et al., Neural-symbolic learning and reasoning: A survey and interpretation 1, in: *Neuro-Symbolic Artificial Intelligence: The State of the Art*, IOS press, 2021, pp. 1–51.

[6] J. Betker, G. Goh, L. Jing, T. Brooks, J. Wang, L. Li, L. Ouyang, J. Zhuang, J. Lee, Y. Guo et al., Improving image generation with better captions, *Computer Science* **2**(3) (2023), 8.

[7] A. Breit, L. Waltersdorfer, F.J. Ekaputra, M. Sabou, A. Ekelhart, A. Iana, H. Paulheim, J. Portisch, A. Revenko, A.t. Teije et al., Combining machine learning and semantic web: A systematic mapping study, *ACM Computing Surveys* (2023).

[8] T. Brown, B. Mann, N. Ryder, M. Subbiah, J.D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell et al., Language models are few-shot learners, *Advances in neural information processing systems* **33** (2020), 1877–1901.

[9] A. Cattan, A. Eirew, G. Stanovsky, M. Joshi and I. Dagan, Cross-document Coreference Resolution over Predicted Mentions, in: *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, C. Zong, F. Xia, W. Li and R. Navigli, eds, Association for Computational Linguistics, Online, 2021, pp. 5100–5107. doi:10.18653/v1/2021.findings-acl.453. https://aclanthology.org/2021.findings-acl.453.

[10] H. Chen, F. Jiao, X. Li, C. Qin, M. Ravaut, R. Zhao, C. Xiong and S. Joty, ChatGPT's One-year Anniversary: Are Open-Source Large Language Models Catching up?, *arXiv:2311.16989* (2023).

[11] Z. Chen, R. Sun, W. Liu, Y. Hong and C. Gan, Genome: generative neuro-symbolic visual reasoning by growing and reusing modules, *arXiv preprint arXiv:2311.04901* (2023).

[12] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H.W. Chung, C. Sutton, S. Gehrmann et al., Palm: Scaling language modeling with pathways, *Journal of Machine Learning Research* **24**(240) (2023), 1–113.

[13] H.W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, Y. Li, X. Wang, M. Dehghani, S. Brahma et al., Scaling instruction-finetuned language models, *arXiv:2210.11416* (2022).

[14] B.C. Colelough and W. Regli, Neuro-Symbolic AI in 2024: A Systematic Review, in: *Proceedings of the First International Workshop on Logical Foundations of Neuro-Symbolic AI (LNSAI 2024)*, Co-located with IJCAI 2024, Jeju, South Korea, 2024.

[15] P. Colon-Hernandez, C. Havasi, J. Alonso, M. Huggins and C. Breazeal, Combining pre-trained language models and structured knowledge, *arXiv preprint arXiv:2101.12294* (2021).

[16] R. Cordeschi, AI turns fifty: revisiting its origins, *Applied Artificial Intelligence* **21**(4–5) (2007), 259–279.

[17] M. de Boer, Q. Smit, M. van Bekkum, A. Meyer-Vitali and T. Schmid, Modular Design Patterns for Generative Neuro-Symbolic Systems, *GeNeSy* (2024).

[18] L. De Raedt, S. Dumančić, R. Manhaeve and G. Marra, From statistical relational to neuro-symbolic artificial intelligence, *arXiv preprint arXiv:2003.08316* (2020).

[19] J. Devlin, Bert: Pre-training of deep bidirectional transformers for language understanding, *arXiv preprint arXiv:1810.04805* (2018).

[20] A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan et al., The llama 3 herd of models, *arXiv preprint arXiv:2407.21783* (2024).
[21] A. Ellis, B. Dave, H. Salehi, S. Ganapathy and C. Shimizu, EASY-AI: sEmantic And compoSable glYphs for representing AI systems, in: *HHAI 2024: Hybrid Human AI Systems for the Social Good*, IOS Press, 2024, pp. 105–113.
[22] A. Ellis, B. Dave, H. Salehi, S. Ganapathy and C. Shimizu, Implementing SNOOP-AI in CoModIDE, in: *NAECON 2024-IEEE National Aerospace and Electronics Conference*, IEEE, 2024, pp. 101–104.
[23] C. Feng, X. Zhang and Z. Fei, Knowledge solver: Teaching llms to search for domain knowledge from knowledge graphs, *arXiv preprint arXiv:2309.03118* (2023).
[24] Y. Feng, X. Chen, B.Y. Lin, P. Wang, J. Yan and X. Ren, Scalable Multi-Hop Relational Reasoning for Knowledge-Aware Question Answering, in: *Proceedings of EMNLP*, Association for Computational Linguistics, Online, 2020, pp. 1295–1309.
[25] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun and H. Wang, Retrieval-augmented generation for large language models: A survey, *arXiv preprint arXiv:2312.10997* (2023).
[26] A.d. Garcez and L.C. Lamb, Neurosymbolic AI: The 3 rd wave, *Artificial Intelligence Review* **56**(11) (2023), 12387–12406.
[27] B. He, D. Zhou, J. Xiao, X. Jiang, Q. Liu, N.J. Yuan and T. Xu, BERT-MK: Integrating Graph Contextualized Knowledge into Pre-trained Language Models, in: *Findings of the Association for Computational Linguistics: EMNLP 2020*, T. Cohn, Y. He and Y. Liu, eds, Association for Computational Linguistics, Online, 2020, pp. 2281–2290. doi:10.18653/v1/2020.findings-emnlp.207. https://aclanthology.org/2020.findings-emnlp.207.
[28] M. Hilario, An Overview of Strategies for Neurosymbolic Integration, in: *Computational Architectures Integrating Symbolic and Neural Processes*, R. Sun and L. Bookman, eds, Kluwer Academic Publishers, 1994.
[29] P. Hitzler, A. Eberhart, M. Ebrahimi, M.K. Sarker and L. Zhou, Neuro-symbolic approaches in artificial intelligence, *National Science Review* **9**(6) (2022), nwac035. doi:10.1093/nsr/nwac035.
[30] N. Hu, Y. Wu, G. Qi, D. Min, J. Chen, J.Z. Pan and Z. Ali, An empirical study of pre-trained language models in simple knowledge graph question answering, *World Wide Web* **26**(5) (2023), 2855–2886–. doi:10.1007/s11280-023-01166-y.
[31] N. Huang, Y.R. Deshpande, Y. Liu, H. Alberts, K. Cho, C. Vania and I. Calixto, Endowing language models with multimodal knowledge graph representations, *arXiv preprint arXiv:2206.13163* (2022).
[32] X. Huang, W. Ruan, W. Huang, G. Jin, Y. Dong, C. Wu, S. Bensalem, R. Mu, Y. Qi, X. Zhao, K. Cai, Y. Zhang, S. Wu, P. Xu, D. Wu, A. Freitas and M.A. Mustafa, A survey of safety and trustworthiness of large language models through the lens of verification and validation, *Artificial Intelligence Review* **57**(7) (2024), 175. doi:10.1007/s10462-024-10824-0.
[33] S. Ji, S. Pan, E. Cambria, P. Marttinen and S.Y. Philip, A survey on knowledge graphs: Representation, acquisition, and applications, *IEEE transactions on neural networks and learning systems* **33**(2) (2021), 494–514.
[34] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y.J. Bang, A. Madotto and P. Fung, Survey of Hallucination in Natural Language Generation, *ACM Comput. Surv.* **55**(12) (2023).
[35] Y. Jin, J. Li, Y. Liu, T. Gu, K. Wu, Z. Jiang, M. He, B. Zhao, X. Tan, Z. Gan et al., Efficient multimodal large language models: A survey, *arXiv preprint arXiv:2405.10739* (2024).
[36] M. Johansson and N. Smallbone, Exploring mathematical conjecturing with large language models, *Proceedings of NeSy* (2023).
[37] M. Joshi, D. Chen, Y. Liu, D.S. Weld, L. Zettlemoyer and O. Levy, Spanbert: Improving pre-training by representing and predicting spans, *Transactions of the association for computational linguistics* **8** (2020), 64–77.
[38] J. Kaddour, J. Harris, M. Mozes, H. Bradley, R. Raileanu and R. McHardy, Challenges and applications of large language models, *arXiv preprint arXiv:2307.10169* (2023).
[39] P. Ke, H. Ji, Y. Ran, X. Cui, L. Wang, L. Song, X. Zhu and M. Huang, JointGT: Graph-Text Joint Representation Learning for Text Generation from Knowledge Graphs, in: *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, C. Zong, F. Xia, W. Li and R. Navigli, eds, Association for Computational Linguistics, Online, 2021, pp. 2526–2538. doi:10.18653/v1/2021.findings-acl.223. https://aclanthology.org/2021.findings-acl.223.
[40] N.Y. Khanday and S.A. Sofi, Taxonomy, state-of-the-art, challenges and applications of visual understanding: A review, *Computer Science Review* **40** (2021), 100374. doi:https://doi.org/10.1016/j.cosrev.2021.100374. https://www.sciencedirect.com/science/article/pii/S1574013721000149.
[41] B. Kim, T. Hong, Y. Ko and J. Seo, Multi-Task Learning for Knowledge Graph Completion with Pre-trained Language Models, in: *Proceedings of the 28th International Conference on Computational Linguistics*, D. Scott, N. Bel and C. Zong, eds, International Committee on Computational Linguistics, Barcelona, Spain (Online), 2020, pp. 1737–1743. doi:10.18653/v1/2020.coling-main.153. https://aclanthology.org/2020.coling-main.153.
[42] S. Kukreja, T. Kumar, A. Purohit, A. Dasgupta and D. Guha, A Literature Survey on Open Source Large Language Models, in: *Proceedings of the 2024 7th International Conference on Computers in Management and Business*, Association for Computing Machinery, New York, NY, USA, 2024, pp. 133–143–. ISBN 9798400716652.
[43] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel et al., Retrieval-augmented generation for knowledge-intensive nlp tasks, *Advances in Neural Information Processing Systems* **33** (2020), 9459–9474.
[44] S. Li, X. Li, L. Shang, C. Sun, B. Liu, Z. Ji, X. Jiang and Q. Liu, Pre-training language models with deterministic factual knowledge, *arXiv preprint arXiv:2210.11165* (2022).
[45] S. Li, Y. Gao, H. Jiang, Q. Yin, Z. Li, X. Yan, C. Zhang and B. Yin, Graph reasoning for question answering with triplet retrieval, *arXiv preprint arXiv:2305.18742* (2023).

[46] B.Y. Lin, X. Chen, J. Chen and X. Ren, KagNet: Knowledge-Aware Graph Networks for Commonsense Reasoning, in: *Proceedings of EMNLP-IJCNLP*, K. Inui, J. Jiang, V. Ng and X. Wan, eds, Association for Computational Linguistics, Hong Kong, China, 2019, pp. 2829–2839.

[47] Z. Lin, S. Guan, W. Zhang, H. Zhang, Y. Li and H. Zhang, Towards trustworthy LLMs: a review on debiasing and dehallucinating in large language models, *Artificial Intelligence Review* **57**(9) (2024), 243.

[48] D. Lukovnikov, A. Fischer and J. Lehmann, Pretrained Transformers for Simple Question Answering over Knowledge Graphs, in: *The Semantic Web – ISWC 2019: 18th International Semantic Web Conference, Auckland, New Zealand, October 26–30, 2019, Proceedings, Part I*, 2019, pp. 470–486–. ISBN 978-3-030-30792-9.

[49] L. Luo, J. Ju, B. Xiong, Y.-F. Li, G. Haffari and S. Pan, Chatrule: Mining logical rules with large language models for knowledge graph reasoning, *arXiv preprint arXiv:2309.01538* (2023).

[50] X. Lv, Y. Lin, Y. Cao, L. Hou, J. Li, Z. Liu, P. Li and J. Zhou, Do Pre-trained Models Benefit Knowledge Graph Completion? A Reliable Evaluation and a Reasonable Approach, in: *Findings of the Association for Computational Linguistics: ACL 2022*, S. Muresan, P. Nakov and A. Villavicencio, eds, Association for Computational Linguistics, Dublin, Ireland, 2022, pp. 3570–3581. doi:10.18653/v1/2022.findings-acl.282. https://aclanthology.org/2022.findings-acl.282.

[51] A. Martin, AAAI-MAKE 2023: Challenges requiring the combination of machine learning and knowledge engineering, *AI magazine* **44**(2) (2023), 204–205.

[52] K. McGarry, S. Wermter and J. MacIntyre, Hybrid neural systems: from simple coupling to fully integrated neural networks, *Neural Computing Surveys* **2**(1) (1999), 62–93.

[53] L.R. Medsker, *Hybrid Neural Network and Expert Systems*, Kluwer Academic Publishers, Boston, 1994.

[54] A. Meyer-Vitali, W. Mulder and M.H. de Boer, Modular design patterns for hybrid actors, *arXiv preprint arXiv:2109.09331* (2021).

[55] B. Min, H. Ross, E. Sulem, A.P.B. Veyseh, T.H. Nguyen, O. Sainz, E. Agirre, I. Heintz and D. Roth, Recent advances in natural language processing via large pre-trained language models: A survey, *ACM Computing Surveys* **56**(2) (2023), 1–40.

[56] S. Minaee, T. Mikolov, N. Nikzad, M. Chenaghlu, R. Socher, X. Amatriain and J. Gao, Large language models: A survey, *arXiv preprint arXiv:2402.06196* (2024).

[57] M. Minsky, Steps toward artificial intelligence, *Proceedings of the IRE* **49**(1) (1961), 8–30.

[58] T. Mossakowski, Modular design patterns for neural-symbolic integration: refinement and combination, *arXiv:2206.04724* (2022).

[59] N. Muennighoff, T. Wang, L. Sutawika, A. Roberts, S. Biderman, T.L. Scao, M.S. Bari, S. Shen, Z.-X. Yong, H. Schoelkopf et al., Crosslingual generalization through multitask finetuning, *arXiv:2211.01786* (2022).

[60] M. Nayyeri, Z. Wang, M.M. Akter, M.M. Alam, M.R.A.H. Rony, J. Lehmann and S. Staab, Integrating Knowledge Graph Embeddings and Pre-trained Language Models in Hypercomplex Spaces, in: *International Semantic Web Conference*, Springer, 2023, pp. 388–407.

[61] OECD, *Is Education Losing the Race with Technology?: AI's Progress in Maths and Reading*, Educational Research and Innovation, OECD Publishing, Paris, 2023. doi:10.1787/73105f99-en.

[62] L. Pan, A. Albalak, X. Wang and W.Y. Wang, Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning, *arXiv:2305.12295* (2023).

[63] S. Pan, L. Luo, Y. Wang, C. Chen, J. Wang and X. Wu, Unifying large language models and knowledge graphs: A roadmap, *IEEE Transactions on Knowledge and Data Engineering* (2024).

[64] S. Park and H. Kim, Improving sentence-level relation extraction through curriculum learning, *arXiv preprint arXiv:2107.09332* (2021).

[65] B. Peng, E. Alcaide, Q. Anthony, A. Albalak, S. Arcadinho, S. Biderman, H. Cao, X. Cheng, M. Chung, M. Grella et al., Rwkv: Reinventing rnns for the transformer era, *arXiv preprint arXiv:2305.13048* (2023).

[66] M.E. Peters, M. Neumann, R.L. Logan IV, R. Schwartz, V. Joshi, S. Singh and N.A. Smith, Knowledge enhanced contextual word representations, *arXiv:1909.04164* (2019).

[67] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever et al., Language models are unsupervised multitask learners, *OpenAI blog* **1**(8) (2019), 9.

[68] R. Rombach, A. Blattmann, D. Lorenz, P. Esser and B. Ommer, High-Resolution Image Synthesis with Latent Diffusion Models, 2021.

[69] C. Rosset, C. Xiong, M. Phan, X. Song, P. Bennett and S. Tiwary, Knowledge-aware language model pretraining, *arXiv preprint arXiv:2007.00655* (2020).

[70] G. Rossiello, M.F.M. Chowdhury, N. Mihindukulasooriya, O. Cornec and A.M. Gliozzo, KnowGL: Knowledge Generation and Linking from Text, in: *AAAI*, 2023, pp. 16476–16478.

[71] M. Sabou, M. Llugiqi, F.J. Ekaputra, L. Waltersdorfer and S. Tsaneva, Knowledge engineering in the age of neurosymbolic systems, *Neurosymbolic AI Journal (under review)* (2024).

[72] S. Samsi, D. Zhao, J. McDonald, B. Li, A. Michaleas, M. Jones, W. Bergeron, J. Kepner, D. Tiwari and V. Gadepally, From words to watts: Benchmarking the energy costs of large language model inference, in: *2023 IEEE High Performance Extreme Computing Conference (HPEC)*, IEEE, 2023, pp. 1–9.

[73] N. Savage, The race to the top among the world's leaders in artificial intelligence, *Nature* **588**(7837) (2020), S102–S102.

[74] T. Schmid, A Systematic and Efficient Approach to the Design of Modular Hybrid AI Systems, in: *AAAI Spring Symposium on Challenges Requiring the Combination of Machine Learning and Knowledge Engineering (AAAI-MAKE)*, CEUR Workshop Proceedings, Vol. 3433, 2023.

[75] W.J. Schmidt, D. Rincon-Yanez, E. Kharlamov and A. Paschke, Scaling Scientific Knowledge Discovery with Neuro-Symbolic AI and Large Language Models, in: *Proceedings of the First International Workshop on Scaling Knowledge Graphs for Industry, co-located with the 20th International Conference on Semantic Systems (SEMANTICS)*, Amsterdam, Netherlands, 2024.

[76] R. Schwartz, J. Dodge, N.A. Smith and O. Etzioni, Green ai, *Communications of the ACM* **63**(12) (2020), 54–63.

[77] T. Shen, Y. Mao, P. He, G. Long, A. Trischler and W. Chen, Exploiting structured knowledge in text via graph-guided representation learning, *arXiv preprint arXiv:2004.14224* (2020).
[78] P. Shi and J. Lin, Simple bert models for relation extraction and semantic role labeling, *arXiv preprint arXiv:1904.05255* (2019).
[79] Y. Su, X. Han, Z. Zhang, Y. Lin, P. Li, Z. Liu, J. Zhou and M. Sun, CokeBERT: Contextual knowledge selection and embedding towards enhanced pre-trained language models, *AI Open* **2** (2021), 127–134.
[80] J. Sun, C. Xu, L. Tang, S. Wang, C. Lin, Y. Gong, H.-Y. Shum and J. Guo, Think-on-Graph: Deep and Responsible Reasoning of Large Language Model on Knowledge Graph, 2023.
[81] R. Sun and F. Alexandre, *Connectionist-Symbolic Integration: From Unified to Hybrid Approaches*, 1st edn, Psychology Press, 1997.
[82] Y. Sun, Q. Shi, L. Qi and Y. Zhang, JointLK: Joint Reasoning with Language Models and Knowledge Graphs for Commonsense Question Answering, 2022, pp. 5049–5060.
[83] H. Tian, C. Gao, X. Xiao, H. Liu, B. He, H. Wu, H. Wang and F. Wu, SKEP: Sentiment knowledge enhanced pre-training for sentiment analysis, *arXiv preprint arXiv:2005.05635* (2020).
[84] M. van Bekkum, M. de Boer, F. van Harmelen, A. Meyer-Vitali and A.t. Teije, Modular design patterns for hybrid learning and reasoning systems: a taxonomy, patterns and use cases, *Applied Intelligence* **51**(9) (2021), 6528–6546.
[85] F. van Harmelen, Preface: The 3rd AI wave is coming, and it needs a theory, in: *Neuro-Symbolic Artificial Intelligence: The State of the Art*, P. Hitzler and M.K. Sarker, eds, IOS Press, 2022, p. V–VII. doi:10.3233/FAIA210347-fm.
[86] F. Van Harmelen and A. Ten Teije, A boxology of design patterns for hybrid learning and reasoning systems, *Journal of Web Engineering* **18**(1–3) (2019), 97–123.
[87] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser and I. Polosukhin, Attention is all you need, *Advances in neural information processing systems* **30** (2017).
[88] J. Wang, Q. Sun, X. Li and M. Gao, Boosting language models reasoning with chain-of-knowledge prompting, *arXiv preprint arXiv:2306.06427* (2023).
[89] J. Wang, H. Jiang, Y. Liu, C. Ma, X. Zhang, Y. Pan, M. Liu, P. Gu, S. Xia, W. Li et al., A Comprehensive Review of Multimodal Large Language Models: Performance and Challenges Across Different Tasks, *arXiv preprint arXiv:2408.01319* (2024).
[90] P. Wang, X. Xie, X. Wang and N. Zhang, Reasoning through memorization: Nearest neighbor knowledge graph embeddings, in: *CCF International Conference on Natural Language Processing and Chinese Computing*, Springer, 2023, pp. 111–122.
[91] X. Wang, T. Gao, Z. Zhu, Z. Zhang, Z. Liu, J. Li and J. Tang, KEPLER: A unified model for knowledge embedding and pre-trained language representation, *Transactions of the Association for Computational Linguistics* **9** (2021), 176–194.
[92] X. Wang, Q. He, J. Liang and Y. Xiao, Language models as knowledge embeddings, *arXiv preprint arXiv:2206.12617* (2022).
[93] Y. Wang, N. Lipka, R. Rossi, A. Siu, R. Zhang and T. Derr, Knowledge Graph Prompting for Multi-Document Question Answering, *Proceedings of the AAAI Conference on Artificial Intelligence* **38** (2024), 19206–19214.
[94] X. Wei, S. Wang, D. Zhang, P. Bhatia and A. Arnold, Knowledge enhanced pretrained language models: A compreshensive survey, *arXiv:2110.08455* (2021).
[95] Y. Wen, Z. Wang and J. Sun, Mindmap: Knowledge graph prompting sparks graph of thoughts in large language models, *arXiv preprint arXiv:2308.09729* (2023).
[96] H.F. Witschel, B. Barroca, A. Correia, A. Martin and C. Caldera, Visualization of Patterns for Hybrid Learning and Reasoning with Human Involvement, in: *Proceedings of the 12th International Conference on Agents and Artificial Intelligence*, SCITEPRESS, 2020, pp. 423–430. doi:10.5220/0008975504230430.
[97] X. Xie, Z. Li, X. Wang, Z. Xi and N. Zhang, Lambdakg: A library for pre-trained language model-based knowledge graph embeddings, *arXiv preprint arXiv:2210.00305* (2022).
[98] W. Xiong, J. Du, W.Y. Wang and V. Stoyanov, Pretrained encyclopedia: Weakly supervised knowledge-pretrained language model, *arXiv preprint arXiv:1912.09637* (2019).
[99] H. Yan, T. Gui, J. Dai, Q. Guo, Z. Zhang and X. Qiu, A Unified Generative Framework for Various NER Subtasks, in: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, C. Zong, F. Xia, W. Li and R. Navigli, eds, Association for Computational Linguistics, Online, 2021, pp. 5808–5822. doi:10.18653/v1/2021.acl-long.451. https://aclanthology.org/2021.acl-long.451.
[100] S. Yang, X. Li, L. Cui, L. Bing and W. Lam, Neuro-Symbolic Integration Brings Causal and Reliable Reasoning Proofs, 2023.
[101] L. Yao, C. Mao and Y. Luo, KG-BERT: BERT for knowledge graph completion, *arXiv preprint arXiv:1909.03193* (2019).
[102] M. Yasunaga, H. Ren, A. Bosselut, P. Liang and J. Leskovec, QA-GNN: Reasoning with Language Models and Knowledge Graphs for Question Answering, in: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Association for Computational Linguistics, 2021, pp. 535–546.
[103] M. Yasunaga, A. Bosselut, H. Ren, X. Zhang, C.D. Manning, P.S. Liang and J. Leskovec, Deep bidirectional language-knowledge graph pretraining, *Advances in Neural Information Processing Systems* **35** (2022), 37309–37323.
[104] D. Yu, C. Zhu, Y. Yang and M. Zeng, Jaket: Joint pre-training of knowledge graph and language understanding, in: *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36, 2022, pp. 11630–11638.
[105] D. Yu, B. Yang, D. Liu and H. Wang, A Survey on Neural-symbolic Learning Systems, 2021.
[106] D. Zhang, Y. Yu, C. Li, J. Dong, D. Su, C. Chu and D. Yu, Mm-llms: Recent advances in multimodal large language models, *arXiv preprint arXiv:2401.13601* (2024).
[107] S. Zhang, L. Dong, X. Li, S. Zhang, X. Sun, S. Wang, J. Li, R. Hu, T. Zhang, F. Wu et al., Instruction tuning for large language models: A survey, *arXiv preprint arXiv:2308.10792* (2023).

[108] X. Zhang, A. Bosselut, M. Yasunaga, H. Ren, P. Liang, C.D. Manning and J. Leskovec, GreaseLM: Graph reasoning enhanced language models for question answering, *arXiv preprint arXiv:2201.08860* (2022).
[109] Z. Zhang, X. Han, Z. Liu, X. Jiang, M. Sun and Q. Liu, ERNIE: Enhanced language representation with informative entities, *arXiv preprint arXiv:1905.07129* (2019).
[110] Z. Zhang, X. Liu, Y. Zhang, Q. Su, X. Sun and B. He, Pretrain-KGE: Learning Knowledge Representation from Pretrained Language Models, in: *Findings of the Association for Computational Linguistics: EMNLP 2020*, T. Cohn, Y. He and Y. Liu, eds, Association for Computational Linguistics, Online, 2020, pp. 259–266. doi:10.18653/v1/2020.findings-emnlp.25. https://aclanthology.org/2020.findings-emnlp.25.
[111] H. Zhao, H. Chen, F. Yang, N. Liu, H. Deng, H. Cai, S. Wang, D. Yin and M. Du, Explainability for large language models: A survey, *ACM Transactions on Intelligent Systems and Technology* **15**(2) (2024), 1–38.
[112] W.X. Zhao et al., A Survey of Large Language Models, *arXiv preprint arXiv:2303.18223* (2023). https://arxiv.org/abs/2303.18223.
[113] W. Zhong, R. Cui, Y. Guo, Y. Liang, S. Lu, Y. Wang, A. Saied, W. Chen and N. Duan, Agieval: A human-centric benchmark for evaluating foundation models, *arXiv preprint arXiv:2304.06364* (2023).