Translating Requirements in Property Specification Patterns using LLMs

Dario Guidotti[1,*], Laura Pandolfo[1], Tiziana Fanni[2], Katiuscia Zedda[2] and Luca Pulina[1]

[1]University of Sassari, Piazza Università 21, Sassari, 07100, Italy
[2]Abinsula Srl, Viale Umberto 42, Sassari, 07100, Italy

Abstract

This paper introduces ReqH, an innovative tool designed to streamline the translation of natural language requirements into Property Specification Patterns. The tool leverages the capabilities of Large Language Models, which are renowned for their ability to comprehend and generate human-like text. ReqH aims to address the challenges of translating informal requirements into formal specifications, a process that is crucial in industrial contexts, particularly within safety and security-critical domains which demand rigorous formalisation to ensure the reliability and security of systems. We present some preliminary results from evaluating our methodology on a dataset of semi-automatically generated automotive requirements. The findings indicate that Large Language Models, when applied to this translation process, show significant potential for improving the accuracy and efficiency of requirement specification.

Keywords

Natural Language Processing, Formal Specifications, Large Language Models, Property Specification Patterns

1. Introduction

In industrial contexts, particularly within safety and security-critical domains, the accurate specification of requirements is of paramount importance. Requirements serve as the foundation upon which systems are designed, developed, and validated. In these high-stakes environments, any ambiguity or error in requirement specification can lead to catastrophic failures, resulting in significant financial losses, harm to human life, or severe environmental damage. Therefore, ensuring that requirements are both correctly captured and precisely translated into formal specifications is essential for the integrity and reliability of systems [2]. Natural language remains the most common medium for expressing requirements due to its accessibility and ease of use by domain experts. However, natural language is inherently ambiguous and often lacks the precision required for formal verification and validation processes. In safety and security-critical domains, formal specifications are crucial because they enable the use of formal methods—mathematically based techniques for the rigorous specification, development, and verification of software and hardware systems. Formal methods allow for the exhaustive verification of system properties, ensuring that critical requirements, such as safety constraints and security protocols, are met without errors or omissions [3]. These methods have been successfully applied across various domains, including the verification of hardware circuits [4, 5], flight control systems [6], and increasingly in machine learning [7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24], where they help ensure the reliability and correctness of complex models. The translation of natural language requirements into formal specifications, such as Property Specification Patterns (PSPs) [25], presents a significant challenge. This task requires both a deep understanding of the domain-specific requirements and expertise in formal languages, which may often be inaccessible to domain experts. As a result, the

AI4CC-IPS-RCRA-SPIRIT 2024: International Workshop on Artificial Intelligence for Climate Change, Italian Workshop on Planning and Scheduling, RCRA Workshop on Experimental evaluation of algorithms for solving problems with combinatorial explosion, and SPIRIT Workshop on Strategies, Prediction, Interaction, and Reasoning in Italy. November 25-28th, 2024, Bolzano, Italy [1].
*Corresponding author.
dguidotti@uniss.it (D. Guidotti); lpandolfo@uniss.it (L. Pandolfo); tiziana.fanni@abinsula.com (T. Fanni); katiuscia.zedda@abinsula.com (K. Zedda); lpulina@uniss.it (L. Pulina)
0000-0001-8284-5266 (D. Guidotti); 0000-0002-5785-5638 (L. Pandolfo); 0000-0002-4301-6497 (T. Fanni); 0009-0003-1059-8261 (K. Zedda); 0000-0003-0258-3222 (L. Pulina)

translation process is prone to errors and inefficiencies, which can compromise the effectiveness of formal verification. This topic was the focus of the Use Case proposed by Abinsula Srl in the scope of the AIDOaRt project, a Key Digital Technologies Joint Undertaking (KTD JU) project started on April 2021, in which the University of Sassari participates as an Artificial Intelligence (AI)-based solutions provider.

To address this challenge, we introduce ReqH, a tool designed to facilitate the translation of natural language requirements into PSP by leveraging Large Language Models (LLMs). LLMs, which have been widely recognised for their ability to understand and generate human-like text, offer a promising solution to bridge the gap between natural language and formal specifications. ReqH aims to simplify the translation process, making it more accessible to domain experts who may not be proficient in formal languages. To evaluate the effectiveness of our proposed methodology, we collaborated with automotive domain experts from Abinsula to develop a comprehensive dataset. This dataset comprises 1000 requirements articulated in natural language, along with their corresponding PSP versions. This dataset serves as a crucial resource for testing and refining ReqH's capabilities, providing a robust foundation for assessing the tool's performance in real-world scenarios. In addition to developing the dataset, we employed another tool, ReqV, to automate the evaluation of the syntactic correctness of the PSP translations produced by ReqH. ReqV systematically checks the translations against predefined syntactic rules, thereby offering an objective measure of the accuracy and reliability of the generated PSPs. This automated evaluation process not only facilitates large-scale testing but also ensures that the results are consistent and reproducible.

The preliminary results from our evaluation highlight ReqH's ability to handle a diverse range of requirements, demonstrating its versatility across different types of specifications. While the tool shows strong potential, the results also indicate areas for further enhancement, particularly in refining the translation accuracy for more complex or ambiguous requirements. Nevertheless, these findings affirm that ReqH has the potential to significantly reduce the manual effort required in the translation process, offering valuable support to domain experts who may not be familiar with formal languages.

The remainder of this paper is organised as follows: Section 2 introduces some basic concepts and definitions. Section 3 present the Abinsula Case Study from which this work originates and, more in general, the AIDOaRt Project. Section 4 presents the methodology employed in our study and the models and dataset considered in our experimental evaluation. Section 5 describes the experimental setup and presents the results of our empirical analysis. Finally, in Section 6, we briefly summarise our conclusions and highlight future research.

2. Background

This section provides an overview of the key concepts underlying our work: Natural Language Processing, Large Language Models, and Property Specification Patterns. These topics form the foundation for understanding the methodology and tools developed in this paper.

2.1. Natural Language Processing

Natural Language Processing (NLP) is a crucial area of artificial intelligence that focuses on enabling computers to understand, interpret, and generate human language. It serves as a bridge between human communication and computer understanding, making it possible for machines to process and interact with language in a meaningful way. NLP encompasses a wide array of tasks, including but not limited to text analysis, sentiment analysis, machine translation, speech recognition, information retrieval, and natural language understanding [26].

The importance of NLP has grown significantly in industrial contexts, especially within safety and security-critical domains such as automotive, aerospace, and healthcare. In these sectors, vast amounts of textual data—such as technical documentation, maintenance logs, and safety regulations—must be processed efficiently and accurately. NLP techniques enable the automation of these processes, improving both speed and accuracy. For instance, in requirement engineering, NLP is used to extract structured

information from unstructured text, identify key requirements, and even detect inconsistencies or ambiguities in the text.

However, despite the advancements in NLP, the field faces significant challenges due to the inherent complexities of natural language. Language is often ambiguous, context-dependent, and varies greatly in structure and vocabulary. These characteristics make it difficult for machines to consistently interpret text in the intended way, particularly in domains where precision and accuracy are critical. For example, a single requirement written in natural language might be interpreted differently by different readers, leading to inconsistencies when translating these requirements into formal specifications. Overcoming these challenges is essential for the effective application of NLP in critical domains, where the stakes are high and the cost of errors can be severe.

2.2. Large Language Models

Large Language Models [27] represent a major breakthrough in the field of Natural Language Processing, marking a significant step forward in the ability of machines to understand and generate human-like text. LLMs, such as OpenAI's GPT (Generative Pre-trained Transformer) series and Google's BERT (Bidirectional Encoder Representations from Transformers), are trained on extensive datasets that include billions of words from diverse sources such as books, articles, and websites. This vast training data enables LLMs to learn the statistical properties of language, including grammar, syntax, semantics, and even some level of contextual understanding.

One of the key strengths of LLMs lies in their ability to perform a wide range of NLP tasks with minimal task-specific training, a capability often referred to as "few-shot" or "zero-shot" learning. This means that LLMs can generalise from a few examples or even tackle tasks they have not been explicitly trained on. This adaptability makes LLMs highly valuable in various applications, including automated content generation, dialogue systems, language translation, summarisation, and more.

In the context of requirement engineering, LLMs offer a promising solution for the complex task of translating natural language requirements into formal specifications, such as Property Specification Patterns. Traditional methods of translation often require deep domain expertise in both the subject matter and formal languages, making the process time-consuming and prone to errors. LLMs, with their advanced language understanding capabilities, can assist in this translation by automatically generating formal specifications from natural language inputs. This not only speeds up the process but also reduces the likelihood of errors, as LLMs can help ensure that the nuances of the original requirements are captured accurately in the formal specification.

The application of LLMs in this domain is particularly valuable in safety and security-critical industries, where the precision of requirement translation directly impacts the reliability and safety of the final system. By leveraging LLMs, we can bridge the gap between the informal language used by domain experts and the formal languages needed for system verification and validation, thereby improving the overall robustness and safety of industrial systems.

2.3. Property Specification Patterns

Property Specification Patterns are a powerful formalism used to express system properties in a structured, standardised, and reusable manner. They are meant to describe the structure of systems' behaviours and provide expressions of such behaviours in a range of common formalisms. PSPs provide a high-level, user-friendly language for specifying behavioural properties of systems, such as safety conditions, liveness, timing constraints, and response requirements. These patterns are designed to be accessible to engineers and system designers, even those who may not be experts in formal methods or temporal logic.

The concept of PSPs was introduced to address the need for a common language that could be used to specify recurring types of properties across different systems. PSPs encapsulate best practices in formal specification, offering predefined templates that can be adapted to various contexts. For example, a safety-critical requirement, such as "The system must never enter an unsafe state," can be expressed

using a standard PSP template, which can then be encoded into a formal language like Linear Temporal Logic (LTL) [28], Computational Tree Logic (CTL) [29] or Graphical Interval Logic (GIL) [30].

In safety and security-critical systems, the use of PSPs is particularly important because they help ensure that requirements are specified in a precise and unambiguous manner, which is crucial for formal verification and validation. Formal methods, enabled by PSPs, allow for the exhaustive analysis of system behaviours to verify that they meet all specified requirements. This is particularly critical in industries such as automotive, aerospace, and medical devices, where even minor errors in system behaviour can have catastrophic consequences.

However, translating natural language requirements into PSPs is a complex and challenging task. It requires not only a deep understanding of the domain-specific requirements but also expertise in formal languages and formal methods. This dual expertise is often rare, leading to a significant bottleneck in the requirement specification process.

3. Abinsula Use Case

The use case presented in this paper is centred around a real-world application proposed by Abinsula, a leading Italian company specialising in embedded systems mainly for the Automotive, Medical, Precise Agriculture and IoT markets. To ensure quality, compliance with the established time limits and performance reliability, Abinsula formalised its embedded software development procedures. An important part of these procedures is related to the SW requirements analysis that can be summarised in the following main steps:

1. Specify the software requirements
2. Structure software requirements
3. Analyse software requirements against verification criteria
4. Establish bidirectional traceability
5. Ensure consistency between stakeholder requirements and software requirements

Abinsula defined a set of templates to execute these procedures, but none of them is automated. The work described in this paper sets an important milestone in the path towards the automation of steps 3 and 5 described in the bullet list above. The specific use case brought by Abinsula involves the development of a virtual rear-view mirror system, leveraging multiple cooperative cameras and AI-based technology to enhance vehicle safety and driver awareness. This system captures the external environment surrounding the vehicle, processing the data in real-time to provide drivers with a comprehensive view. One of the challenges addressed in this case study are the formal verification of the system against predefined specifications and ensuring the predictability and reliability of the AI-based components.

As in all safety-critical domains, the role of requirements is crucial. This is particularly evident in this use case, where the development of a virtual rear-view mirror system by Abinsula necessitates strict adherence to safety standards. Specifically, the ISO 16505:2019 standard, titled "Road Vehicles — Ergonomic and Performance Aspects of Camera Monitor Systems — Requirements and Test Procedures" [31], specifies the requirements for camera-monitor systems in road vehicles. The ISO 16505:2019 standard outlines technical guidelines, such as camera placement, resolution, and real-time performance, to guarantee that the system operates reliably and safely in real-world conditions. This is crucial in the automotive domain where any failure can result in severe consequences for both human life and the environment.

To verify these safety-critical requirements using formal methods, we leveraged PSPs to translate a set of Abinsula's requirements from natural language into formal specifications. The set of Abinsula's requirements consists of 46 atomic requirements, derived from the ISO 16505:2019 standard and specific customer needs. These requirements cover various aspects of the system, including safety constraints, synchronisation, and timing behaviours. The process of formalising these requirements begins with

analysing the natural language descriptions and identifying recurring patterns that reflect common properties, such as safety, synchronisation, or timing constraints.

By mapping the linguistic elements of the natural language requirements to the corresponding patterns in the formal specification language, we developed a catalogue of 40 property specification patterns tailored to this specific domain. However, while PSPs offer a robust framework for translating requirements, the process is not without challenges. To select the appropriate pattern for each requirement, it is necessary a deep understanding of the system's domain, of the behaviours being specified, and of the available patterns. Misidentifying a pattern or failing to capture a specific requirement can lead to incomplete or incorrect formal specifications. The interpretation and application of these patterns can also be complex, particularly when handling ambiguous or highly detailed natural language requirements.

To illustrate the process of translating natural language requirements into formal specifications using PSPs, the following table presents a concrete example from the Abinsula use case. Consider the following requirement from ISO 16505:2019:

> *The field of view of the Camera Monitoring System shall cover at least the field of view required by the national body for conventional mirrors of the same class, both horizontally and vertically.*

This requirement was partitioned into three atomic requirements expressed in controlled natural language. Each was analysed using PSP templates to identify its scope and pattern, and then translated into a formal specification, as detailed in the table below:

Atomic Requirement	**Scope**	**Pattern**	**PSP Requirement**
IF Class EQUAL 1 AND IF default view THEN vertical vision distance MUST be EQUAL or GREATER THAN 60 m behind driver	Global	Occurrence - Universality	Globally, it is always the case that if class = 1 and default_view holds, then vertical_vision_distance >= 60 holds as well.
IF Class EQUAL 3 AND IF default view THEN vertical vision distance MUST be EQUAL or GREATER THAN 20 m behind driver/passenger	Global	Occurrence - Universality	Globally, it is always the case that if class = 3 and default_view holds, then vertical_vision_distance >= 20 holds as well.
IF Class EQUAL 3 AND IF default view AND vertical longitudinal median plane EQUAL 20 m THEN horizontal vision MUST be EQUAL or GREATER THAN 4 m	Global	Occurrence - Universality	Globally, it is always the case that if class = 3 and default_view and vertical_longitudinal_median_plane >= 20 holds, then horizontal_vision >= 4 holds as well.

Table 1
Examples of translating natural language requirements from ISO 16505:2019 into PSPs. Each row illustrates how an atomic requirement is mapped to its corresponding PSP representation, highlighting the scope, and the pattern.

The challenges encountered in this use case, particularly the complexity of translating natural language requirements into formal specifications, highlight the need for tools like ReqH.

4. Materials and Methods

This section outlines the methodology and dataset used in the development and evaluation of ReqH, a tool designed to facilitate the translation of natural language requirements into Property Specification

Patterns. The following subsections provide a detailed explanation of the methodology employed in ReqH and describe the dataset utilised in the evaluation process.

4.1. Tool Description

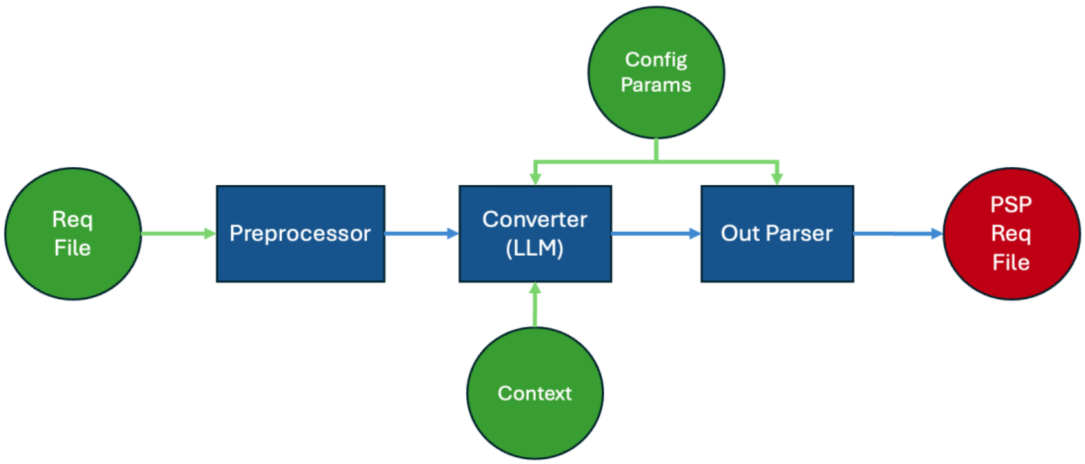

Figure 1: Operational framework of ReqH. The components are represented with blue boxes, the inputs with green circles and the output with a red circle.

ReqH is implemented in Python, a language chosen for its versatility and the availability of robust libraries that facilitate natural language processing and machine learning tasks. The core functionality of ReqH revolves around its ability to process natural language inputs and generate corresponding PSPs, a task traditionally requiring both domain expertise and proficiency in formal methods.

The operational framework of ReqH can be broken down into the following key components:

- **Preprocessor**: The *Preprocessor* is responsible for preparing the input data. It processes the natural language requirements contained in a plain text file, ensuring that they are formatted in a way that is compatible with the subsequent translation steps. This involves standardising the input format, handling any inconsistencies, and ensuring that the text is clean and ready for analysis by the LLM.
- **Converter**: The *Converter* is the core component of ReqH, where the actual translation of requirements into PSPs takes place. This component is tightly integrated with the LangChain library, a powerful tool that provides an interface to various LLMs. LangChain allows users to select the LLM that best suits their needs, offering flexibility in terms of model choice and configuration. During the conversion process, the *Converter* takes into account both the provided context (supplied by the user) and configuration parameters, which guide the LLM in generating the most accurate and relevant PSP translation for each requirement.
- **Out Parser**: After the requirements have been processed by the *Converter*, the results are passed to the *Out Parser*. The *Out Parser* is responsible for formatting the translated PSPs according to the user-defined configuration parameters. This step ensures that the output file, which contains the PSPs, adheres to any specific formatting requirements or standards that the user may have. The final output is a plain text file containing the translated PSPs, ready for use in formal verification and validation processes.

The inputs to ReqH include three main components:

- **Req File**: A plain text file containing the natural language requirements that need to be translated.
- **Context File**: A plain text file containing additional context or guidance that the user wishes to provide to the Converter. This context helps inform the translation process, ensuring that the LLM has the necessary information to generate accurate PSPs.

- **Config Params**: A set of configuration parameters that dictate various options for both the Converter and the Out Parser. These parameters allow users to customise aspects of the translation process, such as the level of detail in the output or specific formatting preferences.

The output of ReqH is a plain text file containing the translated requirements in PSP format. This file is generated after the Out Parser has processed the results from the Converter, ensuring that the PSPs are correctly formatted and ready for use in formal verification processes.

The workflow of ReqH proceeds through a straightforward sequence, aimed at ensuring that natural language requirements are accurately translated into formal specifications. It starts with the user preparing the necessary inputs, which include the natural language requirements and any relevant context. These are provided to ReqH in the form of plain text files, along with configuration parameters that set the specifics for the translation process. Once the inputs are ready, they are processed by the *Preprocessor*, which cleans and standardises the data. This step ensures that the requirements are in a format suitable for translation. Next, the *Converter* uses the chosen LLM to translate each requirement into PSPs. The translation process relies on the context and parameters provided, helping to generate accurate and relevant PSPs. Finally, the translated PSPs go through post-processing by the *Out Parser*, which formats them according to the user's specifications. The end result is a plain text file containing the PSPs, ready for use in formal verification. This structured methodology makes ReqH a practical and effective tool for translating natural language requirements into formal specifications, especially in safety and security-critical areas where precision is crucial.

A key feature of ReqH is its integration with the LangChain library, which significantly enhances the tool's flexibility and adaptability. LangChain provides a seamless interface for building and managing pipelines that involve various Large Language Models (LLMs), enabling users to select and configure different models according to the specific requirements of their task. This flexibility is crucial in safety and security-critical domains, where the precision and reliability of requirement translations are paramount. LangChain offers extensive control over the parameters of LLMs, such as the temperature setting. The temperature parameter controls the balance between creativity and coherence in the model's output, allowing users to adjust the level of randomness in the generated text. This parameterisation ensures that ReqH can be finely tuned to produce coherent output corresponding to the PSP syntaxt. ReqH also leverages Ollama as an execution backend for LLMs to facilitates efficient and scalable model deployment. Ollama provides a robust infrastructure for running LLMs, making it easier to manage computational resources and ensuring that the models operate smoothly during the translation process. For our experimental evaluation, we used the Mistral 7B [1] model, a capable LLM known for handling complex language tasks with good accuracy and efficiency. Mistral's design makes it well-suited for translating detailed requirements in safety-critical domains. While we focused on the Mistral model due to its effectiveness, ReqH is built to easily accommodate different LLMs, allowing it to take advantage of newer and more advanced models as they become available.

Finally, it is important to note that the success of ReqH's translation process is heavily dependent on both the quality of the selected LLM and the appropriateness of the context provided in the prompt. If the chosen model or context is not well-suited to the task, the resulting PSP translations may be flawed. While LLMs like Mistral are highly advanced, they are not infallible, and improper configuration can lead to suboptimal results. Therefore, careful consideration must be given to the selection of the model and the crafting of the prompt to ensure accurate and reliable translations.

4.2. Dataset

In developing a robust dataset for evaluating the ReqH tool, we collaborated closely with domain experts from Abinsula. We began by working with Abinsula's experts to create a set of 40 requirements written in natural language. These requirements were carefully crafted to reflect the typical needs and constraints encountered in the automotive industry, such as those related to vehicle safety systems, communication protocols, and other critical functionalities. The selection of these initial requirements

[1] https://mistral.ai/news/announcing-mistral-7b/

was not arbitrary; it was guided by the intent to cover a wide range of PSPs that are particularly relevant for the automotive use case under consideration.

However, while this initial set of 40 requirements provided a solid foundation, we recognised that it was not sufficient for a thorough validation of the proposed methodology. The automotive domain is vast and complex, and a larger dataset was necessary to adequately assess the effectiveness and reliability of ReqH. Therefore, we decided to expand our dataset significantly. Using the original 40 requirements as a base, we employed a semi-automated approach to generate an additional 1000 requirements, along with their corresponding PSP translations. This expanded dataset was created by varying the original requirements in ways that would generate new, yet syntactically similar, requirements. These variations included changes in specific details, such as parameters or conditions, while maintaining the overall structure and intent of the original requirements. The semi-automated nature of this process allowed us to efficiently generate a large dataset that still reflected the patterns and complexities typical of automotive requirements. It is important to note that, while this expanded dataset provides a much larger sample for evaluation, it may contain requirements with some semantic inconsistencies. These inconsistencies are a natural consequence of the semi-automated generation process and might include minor deviations in meaning or context that were not fully aligned with the original intent of the requirements. However, for the purposes of our current evaluation, these potential semantic issues are not a primary concern. Our focus in this phase of the research is on assessing the syntactical correctness of the PSP translations produced by ReqH. By concentrating on syntax, we can objectively measure how well the tool performs in converting natural language requirements into formal specifications, without being distracted by the more subjective aspects of semantic accuracy.

In summary, the final dataset [32] used in our evaluation consists of 1000 requirements generated through a controlled, semi-automated process. This dataset provides a comprehensive basis for testing the capabilities of ReqH, ensuring that the tool is rigorously evaluated against a wide range of scenarios that are relevant to the automotive industry and beyond.

5. Experimental Evaluation

In this section, we present the experimental evaluation of ReqH, focusing on both the setup and the results of our experiments. The primary goal of this evaluation is to assess the tool's effectiveness in translating natural language requirements into PSP and to verify the syntactical correctness of these translations. We begin by detailing the experimental setup, outlining the process by which the requirements were translated and validated. Following this, we present the results, providing insights into the performance of ReqH and its ability to handle a diverse set of requirements within the automotive domain.

5.1. Experimental Setup

```
You are a language models specialized for the conversion of
   requirements written in Natural Language (NL) to Property
   Specification Patterns (PSP).
Your only output should be the PSP. Do not provide any other output
   except for the requirement in PSP.

In the following you can find some example in the form 'NL':'PSP'.

...
...
...
```

```
Now give me the PSP corresponding to this natural language
    requirement:
```

Listing 1: Example of the context file provided to ReqH during our experimental evaluation.

The requirements from our dataset were sequentially inputted into ReqH. For each requirement, we employed a consistent context file and set of parameters to guide the translation process. This context file served as the initial part of the prompt provided to the LLM used by ReqH, ensuring that the model received consistent guidance throughout the evaluation. The format of the context used is detailed in Listing 1, and it was carefully designed to provide the LLM with the necessary background information to accurately translate the requirements into PSP format. The requirement examples are not reported for the sake of clarity: a total of 18 examples were provided in the original context in the format shown.

For the LLM, we selected the Mistral 7B model, a well-regarded model known for its balance of performance and efficiency in handling complex language tasks. Given the critical nature of the requirements in safety and security-focused domains, it was essential to prioritise the coherence and consistency of the model's outputs. Therefore, the temperature parameter of the LLM was set to 0. The temperature setting in LLMs controls the randomness of the model's responses, with a lower value like 0 reducing variability and encouraging the generation of more predictable, consistent translations. By setting the temperature to 0, we aimed to maximise the coherence of the translations, ensuring that each requirement was processed with a high degree of precision.

After all the requirements were translated using ReqH, the resulting PSPs were subjected to a syntactical validation process. For this purpose, we used ReqV to systematically evaluate each translated PSP, identifying any syntactical errors that may have arisen during the translation process. This validation step was crucial for assessing the reliability of ReqH in producing formally correct specifications that are suitable for use in verification processes.

All the code required to replicate our experiment can be found in the ReqH repository [2].

5.2. Experimental Results

The experimental evaluation of ReqH on our dataset of 1000 requirements yielded a total of 464 successfully translated PSPs. At first glance, this success rate may seem less than ideal, translating less than half of the provided requirements. However, it is essential to place these results within the broader context of the task's complexity.

Translating natural language requirements into formal specifications is inherently challenging due to the nuances and ambiguities that often accompany natural language. Natural language is flexible and expressive, but this flexibility comes at the cost of precision, which is crucial for formal methods. The automotive domain, with its intricate safety and security requirements, exemplifies the kind of environment where this challenge is particularly pronounced. Requirements in this field often involve complex, multi-faceted conditions that can be difficult to capture accurately in formal terms.

The performance of ReqH, while not perfect, highlights its potential as a valuable tool in the requirements engineering process, particularly in safety-critical industries like automotive. The tool successfully handled nearly half of the requirements, demonstrating its capability to assist in the formalisation of specifications. This is significant, considering that even experienced domain experts can struggle with translating requirements into precise formal languages.

Moreover, the results underscore the importance of viewing ReqH as a complementary tool designed to support domain experts rather than replace them. In contexts where accuracy is paramount — such as the automotive industry — ReqH can help alleviate the burden of translating requirements, enabling experts to focus on refining and validating the more complex cases that the tool might not handle perfectly. By automating the translation of a substantial portion of requirements, ReqH can significantly reduce the manual effort required in the formalisation process, streamlining workflows and improving overall efficiency. In this sense, with reference to the SW requirements analysis steps described in

[2] https://github.com/AIMet-Lab/AIDOaRt-UNISS-ReqH

Section 3, the work presented in this paper allows for the potential automation of step 3 (*Analyse software requirements against verification criteria*) and step 5 (*Ensure consistency between stakeholder requirements and software requirements*), therefore counting for the potential automation of 40% of the steps (2 steps over 5).

It is also important to note that the success of ReqH in this evaluation is a foundation upon which future improvements can be built. The results provide valuable insights into the tool's strengths and limitations, offering guidance for further development and fine-tuning. As LLMs continue to advance and as more domain-specific datasets become available, the performance of tools like ReqH is expected to improve, making them even more effective in supporting the needs of safety-critical domains.

6. Conclusions and Future Works

In this paper, we have introduced ReqH, a tool designed to facilitate the translation of natural language requirements into PSPs, leveraging the capabilities of LLMs. Our primary focus was on safety and security-critical domains, particularly within the automotive industry, where the precise formalisation of requirements is essential for ensuring the integrity and reliability of systems. The experimental evaluation demonstrated that ReqH was able to successfully translate 464 out of 1000 requirements into PSPs. While this result indicates that there is room for improvement, it also highlights the inherent complexity of the task. The process of converting natural language—known for its ambiguity and variability—into formal specifications is challenging, particularly in domains that demand high levels of accuracy and precision. Despite these challenges, ReqH has shown promise as a supportive tool for domain experts. It has been proven to be on the right direction to become useful in the company analysis of requirements, which is an important part of the company procedures. In particular, the analysis of the requirements against criteria is time-consuming and error-prone, since it is necessary to check that the whole set of requirements is atomic, unambiguous, correct, complete, consistent and so on. This could lead to the overall automation of 40% of procedures related to requirements analysis. By automating the translation of a significant portion of requirements, ReqH can help reduce the manual effort required in the formalisation process, allowing experts to focus their attention on more complex cases that require human judgement and expertise. This approach not only streamlines the workflow but also enhances the overall efficiency of the requirements engineering process in safety-critical environments.

Looking forward, there are several avenues for future work to enhance the capabilities and performance of ReqH. First, improving the accuracy of the translations will be a key focus. This could involve refining the LLMs used by integrating more advanced models or domain-specific training data. Additionally, incorporating feedback loops where the output of ReqH is continuously evaluated and improved based on expert input could help in progressively enhancing the tool's performance.Finally, future work could explore the integration of ReqH with other tools in the requirements engineering process, creating a more comprehensive ecosystem that supports the entire lifecycle of requirement specification, from natural language capture to formal verification and validation.

In conclusion, while ReqH is still in its early stages, it offers a promising approach to addressing the complex task of translating natural language requirements into formal specifications. With further development and refinement, ReqH has the potential to become an indispensable tool for ensuring the accuracy and reliability of systems in safety and security-critical industries.

Acknowledgments

This research work has received funding through the AIDOaRt project from the ECSEL Joint Undertaking (JU) under grant agreement No 101007350. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Sweden, Austria, Czech Republic, Finland, France, Italy, and Spain.

References

[1] D. Aineto, R. De Benedictis, M. Maratea, M. Mittelmann, G. Monaco, E. Scala, L. Serafini, I. Serina, F. Spegni, E. Tosello, A. Umbrico, M. Vallati (Eds.), Proceedings of the International Workshop on Artificial Intelligence for Climate Change, the Italian workshop on Planning and Scheduling, the RCRA Workshop on Experimental evaluation of algorithms for solving problems with combinatorial explosion, and the Workshop on Strategies, Prediction, Interaction, and Reasoning in Italy (AI4CC-IPS-RCRA-SPIRIT 2024), co-located with 23rd International Conference of the Italian Association for Artificial Intelligence (AIxIA 2024), CEUR Workshop Proceedings, CEUR-WS.org, 2024.

[2] J. Heckmann, S. Shirlaw, An industrial view of requirements engineering and safety, in: G. Rabe (Ed.), 14th International Conference on Computer Safety, Reliability and Security, Safecomp 1995, Belgirate, Italy, October 11-13, 1995, Springer, 1995, pp. 411–416. doi:10.1007/978-1-4471-3054-3_28.

[3] S. Jones, D. Till, A. M. Wrightson, Formal methods and requirements engineering: Challenges and synergies, J. Syst. Softw. 40 (1998) 263–273. doi:10.1016/S0164-1212(97)00171-4.

[4] A. Yasin, T. Su, S. Pillement, M. J. Ciesielski, Formal verification of divider circuits by hardware reduction, in: 19th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design, SMACD 2023, Funchal, Portugal, July 3-5, 2023, IEEE, 2023, pp. 1–4. doi:10.1109/SMACD58065.2023.10192137.

[5] L. Pandolfo, L. Pulina, S. Vuotto, Smt-based consistency checking of configuration-based components specifications, IEEE Access 9 (2021) 83718–83726. doi:10.1109/ACCESS.2021.3085911.

[6] G. Katz, C. W. Barrett, D. L. Dill, K. Julian, M. J. Kochenderfer, Reluplex: An efficient SMT solver for verifying deep neural networks, in: Computer Aided Verification - 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I, volume 10426 of *Lecture Notes in Computer Science*, Springer, 2017, pp. 97–117. doi:10.1007/978-3-319-63387-9_5.

[7] D. Guidotti, L. Pandolfo, L. Pulina, Leveraging satisfiability modulo theory solvers for verification of neural networks in predictive maintenance applications, Inf. 14 (2023) 397. doi:10.3390/INFO14070397.

[8] S. Demarchi, D. Guidotti, A. Pitto, A. Tacchella, Formal verification of neural networks: A case study about adaptive cruise control, in: Proceedings of the 36th ECMS International Conference on Modelling and Simulation, ECMS 2022, Ålesund, Norway, May 30 - June 3, 2022, European Council for Modeling and Simulation, 2022, pp. 310–316. doi:10.7148/2022-0310.

[9] D. Guidotti, Enhancing neural networks through formal verification, in: Discussion and Doctoral Consortium papers of AI*IA 2019 - 18th International Conference of the Italian Association for Artificial Intelligence, Rende, Italy, November 19-22, 2019, volume 2495 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2019, pp. 107–112.

[10] R. Eramo, T. Fanni, D. Guidotti, L. Pandolfo, L. Pulina, K. Zedda, Verification of neural networks: Challenges and perspectives in the aidoart project (short paper), in: Proceedings of the 10th Italian workshop on Planning and Scheduling (IPS 2022), RCRA Incontri E Confronti (RiCeRcA 2022), and the workshop on Strategies, Prediction, Interaction, and Reasoning in Italy (SPIRIT 2022) co-located with 21st International Conference of the Italian Association for Artificial Intelligence (AIxIA 2022), November 28 - December 2, 2022, University of Udine, Udine, Italy, volume 3345 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2022.

[11] D. Guidotti, Safety analysis of deep neural networks, in: Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021, ijcai.org, 2021, pp. 4887–4888. doi:10.24963/IJCAI.2021/675.

[12] D. Guidotti, Verification and repair of neural networks, in: Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021, AAAI Press, 2021, pp. 15714–15715. doi:10.1609/AAAI.V35I18.17854.

[13] D. Guidotti, F. Leofante, L. Pulina, A. Tacchella, Verification and repair of neural networks: A progress report on convolutional models, in: AI*IA 2019 - Advances in Artificial Intelligence - XVIIIth International Conference of the Italian Association for Artificial Intelligence, Rende, Italy, November 19-22, 2019, Proceedings, volume 11946 of *Lecture Notes in Computer Science*, Springer, 2019, pp. 405–417. doi:10.1007/978-3-030-35166-3_29.

[14] D. Guidotti, F. Leofante, L. Pulina, A. Tacchella, Verification of neural networks: Enhancing scalability through pruning, in: ECAI 2020 - 24th European Conference on Artificial Intelligence, 29 August-8 September 2020, Santiago de Compostela, Spain, August 29 - September 8, 2020 - Including 10th Conference on Prestigious Applications of Artificial Intelligence (PAIS 2020), volume 325 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, 2020, pp. 2505–2512. doi:10.3233/FAIA200384.

[15] D. Guidotti, L. Pulina, A. Tacchella, pynever: A framework for learning and verification of neural networks, in: Automated Technology for Verification and Analysis - 19th International Symposium, ATVA 2021, Gold Coast, QLD, Australia, October 18-22, 2021, Proceedings, volume 12971 of *Lecture Notes in Computer Science*, Springer, 2021, pp. 357–363. doi:10.1007/978-3-030-88885-5_23.

[16] S. Demarchi, D. Guidotti, Counter-example guided abstract refinement for verification of neural networks, in: Proceedings of the CPS Summer School PhD Workshop 2022 co-located with 4th Edition of the CPS Summer School (CPS 2022), Pula, Sardinia (Italy), September 19-23, 2022, volume 3252 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2022.

[17] D. Guidotti, Verification of neural networks for safety and security-critical domains, in: Proceedings of the 10th Italian workshop on Planning and Scheduling (IPS 2022), RCRA Incontri E Confronti (RiCeRcA 2022), and the workshop on Strategies, Prediction, Interaction, and Reasoning in Italy (SPIRIT 2022) co-located with 21st International Conference of the Italian Association for Artificial Intelligence (AIxIA 2022), November 28 - December 2, 2022, University of Udine, Udine, Italy, volume 3345 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2022.

[18] D. Guidotti, L. Pandolfo, L. Pulina, Verifying neural networks with non-linear SMT solvers: a short status report, in: 35th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2023, Atlanta, GA, USA, November 6-8, 2023, IEEE, 2023, pp. 423–428. doi:10.1109/ICTAI59109.2023.00068.

[19] D. Guidotti, L. Pandolfo, L. Pulina, Verification of nns in the IMOCO4.E project: Preliminary results, in: 28th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2023, Sinaia, Romania, September 12-15, 2023, IEEE, 2023, pp. 1–4. doi:10.1109/ETFA54631.2023.10275345.

[20] D. Guidotti, L. Pandolfo, L. Pulina, Verifying neural networks with SMT: an experimental evaluation, in: 19th IEEE International Conference on e-Science, e-Science 2023, Limassol, Cyprus, October 9-13, 2023, IEEE, 2023, pp. 1–2. doi:10.1109/E-SCIENCE58273.2023.10254877.

[21] S. Demarchi, D. Guidotti, L. Pulina, A. Tacchella, Supporting standardization of neural networks verification with VNNLIB and coconet, in: Proceedings of the 6th Workshop on Formal Methods for ML-Enabled Autonomous Systems, FoMLAS@CAV 2023, Paris, France, July 17-18, 2023, volume 16 of *Kalpa Publications in Computing*, EasyChair, 2023, pp. 47–58. doi:10.29007/5PDH.

[22] D. Guidotti, L. Pandolfo, L. Pulina, Formal verification of neural networks: A "step zero" approach for vehicle detection, in: Advances and Trends in Artificial Intelligence. Theory and Applications - 37th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2024, Hradec Kralove, Czech Republic, July 10-12, 2024, Proceedings, volume 14748 of *Lecture Notes in Computer Science*, Springer, 2024, pp. 297–309. doi:10.1007/978-981-97-4677-4_25.

[23] D. Guidotti, L. Pandolfo, L. Pulina, Verifying autoencoders for anomaly detection in predictive maintenance, in: Advances and Trends in Artificial Intelligence. Theory and Applications - 37th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2024, Hradec Kralove, Czech Republic, July 10-12, 2024, Proceedings, volume 14748 of *Lecture Notes in Computer Science*, Springer, 2024, pp. 188–199. doi:10.1007/978-981-97-4677-4_16.

[24] D. Guidotti, F. Leofante, A. Tacchella, C. Castellini, Improving reliability of myocontrol using formal verification, IEEE Transactions on Neural Systems and Rehabilitation Engineering 27 (2019) 564–571. doi:10.1109/TNSRE.2019.2893152.

[25] M. B. Dwyer, G. S. Avrunin, J. C. Corbett, Patterns in property specifications for finite-state verification, in: Proceedings of the 1999 International Conference on Software Engineering, ICSE' 99, Los Angeles, CA, USA, May 16-22, 1999, ACM, 1999, pp. 411–420. doi:10.1145/302405.302672.

[26] N. Patwardhan, S. Marrone, C. Sansone, Transformers in the real world: A survey on NLP applications, Inf. 14 (2023) 242. doi:10.3390/INFO14040242.

[27] P. Kumar, Large language models (llms): survey, technical frameworks, and future challenges, Artif. Intell. Rev. 57 (2024) 260. doi:10.1007/S10462-024-10888-Y.

[28] A. Pnueli, Z. Manna, The temporal logic of reactive and concurrent systems, Springer 16 (1992) 12.

[29] E. M. Clarke, E. A. Emerson, A. P. Sistla, Automatic verification of finite-state concurrent systems using temporal logic specifications, ACM Transactions on Programming Languages and Systems (TOPLAS) 8 (1986) 244–263.

[30] L. K. Dillon, G. Kutty, L. E. Moser, P. M. Melliar-Smith, Y. S. Ramakrishna, A graphical interval logic for specifying concurrent systems, ACM Transactions on Software Engineering and Methodology (TOSEM) 3 (1994) 131–165.

[31] I. O. for Standardization, Road vehicles — Ergonomic and performance aspects of Camera Monitor Systems — Requirements and test procedures, ISO 16505:2019 ed., International Organization for Standardization, Vernier, Geneva, Switzerland, 2019. URL: https://www.iso.org/standard/72000.html.

[32] D. Guidotti, L. Pandolfo, L. Pulina, S. Azzena, Automotive Domain Property Specification Pattern Dataset, 2024. doi:10.5281/zenodo.13373276.