Automatic Extraction and Formalization of Temporal Requirements from Text: A Survey

Marisol Barrientos[1][0000−0002−2500−7431], Karolin Winter[2][0009−0003−8030−2964], and Stefanie Rinderle-Ma[1][0000−0001−5656−6108]

[1] Technical University of Munich, Garching, Germany
{marisol.barrientos,stefanie.rinderle-ma}@tum.de
[2] Eindhoven University of Technology, Eindhoven, The Netherlands
k.m.winter@tue.nl

Abstract. Natural Language Processing has opened new paths for business process management and requirements engineering, particularly in automating the extraction and formalization of temporal requirements from diverse documents such as system specifications, legal texts, and business process descriptions. Recently, approaches have been introduced to automate this task, employing various document formats as input and targeting different formal specifications. However, a key challenge persists: effectively comparing these approaches and choosing the most suitable one for a specific task. This paper aims to bridge this research gap by conducting a systematic literature review, including a detailed analysis and comparing existing approaches. This comparison is crucial to determine if the latest Large Language Model-based solutions could surpass existing methods, in effectiveness and ease of use. The systematic literature review enables users to select the most suitable method based on their data and end goals. Moreover, this work proposes the NL2MTL[3] method to bridge some of the gaps identified in the literature analysis, i.e., establishing a comparable assessment method, under-representation of legal texts, poor output context management, and the necessity to automate the formalization of requirements, considering both quantitative and qualitative aspects of time. Addressing the latter aspect, we select Metric Temporal Logic (MTL) as formalization and provide the associated prompts and an evaluation of the NL2MTL output.

Keywords: requirements formalization · natural language processing (nlp) · temporal logic · legal text · business process

1 Introduction

Natural Language Processing (NLP) has been employed for a range of business process management and requirements engineering challenges [66], e.g., conceptual and goal modeling [13, 24], as well as for the formalization of requirements [56]. The recent development of Large Language Models (LLMs) has

[3] https://anonymous.4open.science/r/NL2MTL, DLA: 22.04.2024

enhanced particularly challenging tasks. One of those is the extraction, formalization, and later verification of temporal requirements from documents such as system specifications [6], legal texts [61], and business process descriptions [4,42]. Automating this task becomes vital due to the steadily increasing amounts of documents and the therein described (legal) requirements that companies have to comply with. The large variety of formalizations, e.g., Linear Temporal Logic (LTL), Signal Temporal Logic (STL), or Metric Temporal Logic (MTL), has led to numerous automated requirements extraction methods. This abundance of automation options often leaves domain experts puzzled about the most suitable approach for their specific problem.

Driven by challenges in accurately converting natural language temporal requirements to formal specifications, this study addresses the adequacy of current methods, particularly in capturing qualitative (event order relations) and quantitative (measurable intervals) time aspects across various domains. Consequently, this research pivots around the following three research questions.

RQ1: What mechanisms exist for the (semi-)automatic translation of temporal requirements into formal specifications?

RQ2: To what extent can temporal requirements be represented by formal specifications in the different application domains?

RQ3: How can temporal requirements, including both quantitative and qualitative aspects, be effectively formalized from textual inputs?

RQ1 is explored in Sect. 2 and 3 through a systematic literature review (SLR), where papers are classified by their goal. Each category contains an analysis of the methodologies used, input formats, and application domain (i.e., domain-specific vs general). For addressing RQ2, in Sect. 4 the findings emerging from the systematic literature review are presented. To address RQ3, in Sect. 5 we introduce the NL2MTL method, utilizing a Large Language Model, i.e., GPT-4, for processing legal texts into MTL formulations. This is evaluated following the four key principles of a *good formalization* [45], i.e., correctness (i.e., detecting all atomic propositions and formalizing them following MTL semantics), transparency (i.e., defining all used elements, and indicating the reasoning behind a formal specification), comprehensibility (i.e., easy to interpret), and support for multiple interpretations. This section is followed by a discussion Sect. 6 and conclusions (c.f., Sect. 7).

2 Systematic Literature Review Methodology

The Systematic Literature Review (SLR) follows [33] and includes five phases. References are provided in the supplementary material[3].

Phase 1 - Strategy Planning. The **selected databases** for the literature search comprise ACM, IEEE, Scopus, and Springer. For each of those, we carried out our initial search by executing our **search string** which looks as follows:

```
("natural language" OR "large language model") AND
("temporal logic // requirement" OR "ltl" OR"computational tree logic" OR
"requirement formalization // formalisation // extraction" OR
"formal requirement // specification // verification" OR "automate formalization // formalisation")
```

The search string is created by combining a keyword from the field of natural language processing with another from formal methods, specifically those used for formalizing natural language, e.g., a requirement, into temporal logic. After, we defined the following **inclusion (IC)** and **exclusion criteria (EC)**. The inclusion criteria were defined along with the research questions, but this was not the case for the exclusion criteria. Papers on causality analysis (*EC2*) were excluded, as their focus was quantifying time uncertainty rather than automatically extracting formalizations. In this SLR, we excluded research on automated code extraction (e.g., Python script generation), blockchain, Unified Modeling Language (UML) diagrams, and Attribute-Based Access Control (ABAC) (*EC4*), as our focus was not on these technologies or frameworks, but rather on formal methods or models which support time formalization in a business process.

Table 1. Inclusion and Exclusion Criteria for Selecting Research Papers

ID	Definition
IC1	Publicly accessible, in English, peer-reviewed
IC2	Preference for journal versions over conference papers
IC3	Published from January 1, 2018, to April 1, 2024, focusing on recent developments
IC4	Must detail input/output formats, methodology, and evaluations
IC5	Automates formalization and modeling of temporal requirements using AI
IC6	Automates the augmentation of formal specifications using AI
EC1	Previews, abstracts, and theses
EC2	Causality work (e.g., temporal dependency extraction)
EC3	Event log augmentation
EC4	Papers on automatic extraction of code, a blockchain, UML diagrams, and ABAC
EC5	Research tied to a company, e.g., Hanfor [43] or FRETish [18]

Phase 2 and 3 - Initial and Full-Text Screening. Table 2 summarizes the number of selected papers per database and phase. *Phase 1* contains all papers whose title, keywords, or abstract hit the search string. In *Phase 2* papers were filtered by title, while in *Phase 3* are filtered by abstract.

Table 2. Number of Selected Papers per Database and SLR Phase

Database	Phase 1	Phase 2	Phase 3
ACM	65	24	5
IEEE	1042	128	5
Scopus	338	112	12
Springer	886	49	6
Total without duplicates	2.331	313	28

Phase 4 - Back and Forth Snowballing. After completing *Phase 3*, we included 28 papers that satisfied the inclusion and exclusion criteria. We then proceeded to *Phase 4*, where backward and forward snowballing through Google Scholar identified 16 additional relevant publications. This increased the total to 44 papers for further analysis.

Phase 5 - Analysis and Classification. The selected papers are classified and analyzed in Sect. 3 summarizes findings and research gaps in Sect. 4.

3 Classification of Selected Papers

In this section, the 44 papers collected as described in Sect. 2 are analyzed and categorized by goal. This leads to the following categories, *Automatic Generation of Formal Specifications* (cf., Sect.3.1), *Automatic Generation of Modeling Notations* (cf., Sect. 3.2), and *Automatic Augmentation of Formal Specifications* (cf., Sect. 3.3). All categories can be further divided based on output formats. Tables 3 - 6 contain an overview of the papers by category, including publication year, input and output format, methodology, application, and URL to their project website.

3.1 Automatic Generation of Formal Specifications

Linear Temporal Logic. [49] used the Stanford Parser to extract LTL formulas from system requirements, addressing ambiguity detection limitations but struggled with complex inputs. [57] furthered this by incorporating smart home IoT knowledge to handle ambiguities. [27] also utilized a similar approach, but their focus was within the robotics domain. In this domain, they are interested in automatically generating LTL formulas to verify tasks, including grounding, navigation, and tasks related to surgical procedures. The works of [7] and [57] both emphasize advancements in robotics aimed at interpreting complex inputs from human and robotic perspectives. Future papers will likely concentrate on robots that can understand human speech in particular contexts, adapting to linguistic and environmental shifts.

Still, in the robotics domain, challenges posed by language complexity and the potential to incorporate context are overcome by approaches that leverage LLMs, as discussed in [48], [37], and [50]. Papers that accept as input format unstructured text [19, 26] and user input [25] also relied on similar methods. While in the ones which had as input format traffic law [20,36], the formalization of constraints was nearly manual.

Signal Temporal Logic. STL is designed for timing events, suitable for applications like cyber-physical systems and smart cities, enabling requirements such as *the soil moisture at any given sensor must remain above a certain threshold throughout the day.* DeepSTL [31] and NL2TL [14] facilitate STL formula extraction. nl2spec [19] provides a conceptual STL extraction extension. CitySpec [16] employs SaSTL for spatial and aggregation considerations in requirements like *the average soil moisture across all sensors in a particular region must remain above a certain threshold for the next 10 days.*

Propositional Projection Temporal Logic. In [58], the PPTLGenerator was introduced, leveraging Stanford CoreNLP[4] for analyzing safety system properties and converting them into PPTL formulas. Following up, [35] presented NL2PPTL, utilizing a Seq2Seq[5] model for converting security requirements into PPTL, significantly enhancing automation with modern ML for complexity management, in contrast to the first approach's reliance on traditional NLP.

[4] https://github.com/stanfordnlp/CoreNLP, DLA: 22.04.2024

[5] https://google.github.io/Seq2Seq/, DLA: 22.04.2024

Table 3. Input-Output Mapping - Automatic Generation of Formal Specifications

Ref.	Year	Input Format	Output Format	Methodology	Application	url
[29]	2018	Requirement	Event-B [Ql: yes, Qt: no]	Model federation	Requirements engineering (landing gear system)	yes
[30]	2022	Unstructured Text	FOL, regex, and LTL [Ql: 1/2, Qt: no]	Fine-tune LLM (T5)	General	no
[49]	2019	Natural Language Requirement	Linear Temporal Logic (LTL) [Ql: yes, Qt: no]	Standford Parser, build dependency tree and map it with the LTL dependency tree	Multiple system requirements (focus on consistency checking)	no
[65]	2020	Natural Language Requirement		Grammar based method	Smart home IoT	no
[57]	2020	Robot command		Semantic parser	Robotics (grounding)	no
[20]	2020	Traffic Law / Code		Manual step	Self-driving vehicles	
[36]	2022			First, converted to constraints on Markov Decision Process (manual step)	Self-driving vehicles	no
[25]	2023	User Input		LLM (GPT-based)	Healthcare process	yes
[27]	2020			Grammar-enhanced one-shot learning synthesis	Robotics	yes
[19]	2023	Unstructured Text		LLMs (Boom and Codex)	General	yes
[26]	2023			LLMs (GPT-3x and GPT-4)	General	no
[7]	2023	Procedural natural language		Part-Of-Speech tagging combined with Semantic Role Labeling	Robotics (surgery)	yes
[48]	2023	Robotic Task		Fine-tune LLM (BART)	Robotics	yes
[37]	2023	Ground Temporal Navigational Commands		Leveraging LLMs (GPT-based and T5) and constructing and training a Seq2Seq transformer model	Robotics (navigation)	yes
[50]	2024	Object goal navigation and mobile pick-and-place instructions		Leveraging LLM, using two-stage in context learning strategy	Robotics (grounding, task verification, and motion planning)	yes
[31]	2022	Requirement	Signal Temporal Logic (STL) [Ql: yes, Qt: 1/2]	Grammar-based generation and transformer-based neural translation technique	Safety-critical cyber-physical systems	yes
[16]	2022			Translation models (Seq2Seq, pre-trained Stanford NER Tagger, Bi-LSTM + CRF, and BERT) and online validation (Bayesian CNN-based)	Smart city	no
[14]	2023			Enrich data with LLMs and manual annotations. Fine-tune T5 (compare with Seq2Seq and GPT-3)	Robotics (circuit, navigation, and grounding)	yes
[19]	2023	Unstructured Text		LLMs	General (system verification)	yes
[54]	2023	Natural Language Command		Bi-RNN	General (planning trajectories)	no

Table 4. Input-Output Mapping - Automatic Generation of Formal Specifications

Ref.	Year	Input Format	Output Format	Methodology	Application	url
[58]	2020	Safety property of self-driving vehicles	Propositional Projection Temporal Logic (PPTL) [Ql: yes, Qt: 1/2]	Stanford CoreNLP, WordNet and JavaCC	Self-driving vehicles (verification of safety properties)	no
[35]	2022	Security Requirement		Neural translation model	Requirements engineering	yes
[63]	2021	Semi-formal Representation Model (RCM)	Metric Temporal Logic (MTL) [Ql: yes, Qt: yes] For [63] also Computational Tree Logic (CTL) [Ql: yes, Qt: no]	Stanford, WordNet, and Prolog	General (system requirements)	yes
[28]	2023	Legal Contract Clauses		Deep learning (3 neural network models)	Legal contract formalization	no
[41]	2023	Legal or planning rules		Semantic Role Labeling and LLMs	Self-driving vehicles (legal or planning rules)	yes
[61]	2024	Traffic Law		Trigger-based hierarchical (manual step)	Self-driving vehicles (monitoring)	yes
[64]	2022	Requirement	Semi-formal Representation Model (RCM) [Ql: yes, Qt: 1/2]	Stanford, WordNet, and Prolog	General (system requirements)	yes
[47]	2023					yes
[6]	2023	Automotive Requirement	Timed Computation Tree Logic (TCTL) [Ql: yes, Qt: yes]	LLM (GPT-J-6B) and OptKATE algorithm	Automotive industry	no

Metric Temporal Logic and Computation Tree Logic. In [62], the Requirement Capture Model (RCM) was introduced to convert system requirements into formal specifications using a blend of formal and semi-formal semantics, enhancing interpretation and conversion to MTL and CTL formulas. Subsequent papers [63] [64] utilized Stanford CoreNLP, WordNet[6], and Prolog to assess RCM, identifying challenges with non-prepositional temporal expressions (e.g., *every hour*, *when something happens*, *immediately after*) and clause order. Further research in publication [47] focused on RCM's ability to revert formalized requirements to natural language, addressing ambiguities and aiding in decision-making during requirement formalization.

In [28], they use MTL to formalize legal contracts, addressing the flexibility issues with input text found in previous RCM-based papers. This technique leverages deep learning and intermediate representations for clarity. They included a step to identify functional requirements due to contract complexities. Meanwhile, [41] also explores legal formalization with a focus on autonomous vehicles, utilizing SRL and LLM, suitable for unseen inputs. Conversely, [61] shares

[6] https://wordnet.princeton.edu/, DLA: 22.04.2024

the domain of self-driving vehicles but concentrates on MTL's role in monitoring rather than the automatic formalization of laws.

Timed Computation Tree Logic Formalization. In [6], a toolkit was developed for enhancing the clarity and consistency of automotive requirements in natural language, utilizing the GPT-J-6B[7] model to transform them into Structured English before formalizing into Timed Computation Tree Logic (TCTL). The TCTL set is transformed into first-order logic to generate a script, which is verified and tested on data from the former Daimler AG.

3.2 Automatic Generation of Modeling Notations

The analysis of unstructured text, requirements, and robotic tasks previously omitted full process descriptions, overlooking temporal complexities. This has led to a shift towards including process descriptions and policies. Table 5 presents the input-to-output mapping for the *Modeling Notation Generation* category.

Business Process Model. The Annotated Textual Descriptions of Processes (ATDP) language presented in [52] and [53] allows the translation of process descriptions to LTL over finite traces (LTLf). Recognizing the challenge of sparse annotated data, in [5], researchers later used the GPT-3[8] model to refine entity and relationship extraction from business processes with minimal examples. In [44], this effort evolved into enhancing a process extraction tool to better recognize entity identities through a sophisticated neural network, streamlining the extraction of information from texts.

Declarative Process Model. In workflow management, Dynamic Condition Response (DCR) graphs represent a visual model evolving beyond traditional declarative approaches like Declare. They allow adaptable task management, extensively used in Danish digital government systems, with further enhancements from tools like the DCR Process Highlighter[9] for model automation and refinement [38, 39]. In contrast, DECLARE models and their extensions have been advanced through approaches like Speech2RuM, which converts spoken input into detailed models, supporting sophisticated constraints as seen in MP-Declare [1, 9]. Additionally, the Declo and C-4PM chatbots aid in creating and mining DPM, with the latter utilizing technologies like Rasa[10] and GPT[8] to enhance model support [21, 25, 26]. These innovations have been particularly impactful in healthcare process management.

Decision Model and Notation, Petri Nets, and Related. In the study by [23] a limitation was that the extraction of DMN from NL required inputs to be concise, clear, and focused solely on one decision, excluding any irrelevant or repetitive information. In contrast, in [51] does tackle ambiguities, yet it still adheres to a rule-based methodology. In their later work, the authors employed deep learning models to extract decision models. However, the authors noted

[7] https://huggingface.co/EleutherAI/gpt-j-6b, DLA: 22.04.2024

[8] https://gpt3demo.com

[9] https://documentation.dcr.design, DLA: 22.04.2024

[10] https://rasa.com/

Table 5. Input-Output Mapping - Automatic Generation of Modeling Notations

Ref.	Year	Input Format	Output Format	Methodology	Application	url
[52]	2019	Process Description	Annotated Textual Description of a Process (ATDP) [Ql: yes, Qt: no]	Machine-readable intermediate language	Compliance, conformance, and model consistency checking	no
[53]	2021					no
[23]	2020	Decision Description	Decision Model [Ql: yes, Qt: 1/2]	NLP pipeline	Decision and dependencies extraction	no
[51]	2021			NLP processing software (FreeLing)	Model extraction	yes
[39]	2021	Textual Artifact	Declarative Process Model	Machine-learning and expert system technique	Business process discovery	yes
[3]	2020	User Input	(DECLARE,	Rules and templates [2]	Support users defining declarative constraints	yes
[1]	2020	Process Description	Dynamic Condition Response (DCR) graph, and Multi-Perspective Declare (MP-Declare)) [Ql: yes, Qt: 1/2]	Extension of the rules and templates from [2]	Support users defining declarative constraints	yes
[38]	2019			NLP module	Support user creation of models	yes
[2]	2019			Rules and templates	Model extraction	yes
[5]	2022	Process Description	Process Element and Relation [Ql: yes, Qt: no]	LLM (GPT-3) and in-context learning	Extraction of process information	yes
[44]	2023			Pretrained end-to-end neural coreference resolution	Extraction of process information and creation of model	yes
[17]	2018	Automobile Requirement	Basic Petri Net [Ql: yes, Qt: no]	NLP and domain-specific ontologies	Requirements engineering (consistency and completeness verification)	no
[4]	2023	Process Description	Temporal Compliance Requirement [Ql: 1/2, Qt: yes]	LLM (GPT-4) supported by three similarity measures (TF-IDF, BERT, and spaCy)	Compliance verification	yes
[42]	2023	Process Description	Resource Compliance Requirement [Ql: 1/2, Qt: no]	Extension of a domain-sensitive temporal tagger (Heideltime)	Compliance verification	yes

limitations including difficulties with coreference resolution, handling synonyms, and a limited dataset that only allowed for a maximum of two levels of decision dependency. For Petri Nets extraction, [17] proved to be effective in the automobile sector, although their tool has been tested with a limited number of case studies and remains inaccessible to the public. Lastly, we find [42], and [4] in both cases they did not formalize natural language but extracted a semi-formalization which was later used for compliance verification.

3.3 Automatic Augmentation of Formal Specifications

Table 6 shows the input-to-output mapping for the *Formal Specification Augmentation* category. In [8], the authors stated that converting formal specifications to natural language had limited scope for further advancement. Lately, however, there has been a growing trend in research about converting formal specifications into natural language. Even though these formal specifications are restricted, they can be challenging for a user to interpret when checking the system, leading to potential misunderstandings. This makes users ignore past formal specifications and create new ones.

Table 6. Input-Output Mapping - Automatic Augmentation of Formal Specifications

Ref.	Year	Input Format	Output Format	Methodology	Application	url
[60]	2023	Structural Logic Expression	Natural Language [Ql: 1/2, Qt: no]	Recursive parsing, Tree-LSTM, GCNs, and a decoder	General	no
[32]	2022	Unstructured Text	Requirement [Ql: 1/2, Qt: 1/2]	Fine-tunning the BERT model	Requirements engineering	no
[29]	2018	Formal Specification	Requirement [Ql: 1/2, Qt: 1/2]	Model Federation	Requirements engineering	yes

In [60], the authors analyze methods for translating logic expressions to natural text using end-to-end Seq2Seq[11] models, which sometimes misinterpret dependencies (e.g., *a motorcycle driver in orange dress*). They also highlight issues with pre-trained language models that introduce noise or invert subjects in logical structures (e.g., *a dog is chasing a cat* instead of *a cat is chasing a dog*). They suggest using structured representations like trees to improve translation accuracy. Additionally, [29] discusses *model federation* to enhance the conversion from natural language to formal specifications, improving traceability and inconsistency analysis. A new approach for extracting requirements using BERT[12], compared against fastText[13] and ELMo[14] baselines, is detailed in [32].

4 Systematic Literature Review Findings

In Sect. 3, the different methods employed to extract formal specifications, models, or to augment natural language were presented, as well as, the application domain of these papers. The following details the findings from the classification, input, and output formats, addressing RQ2.

[11] https://google.github.io/Seq2Seq, DLA: 22.04.2024

[12] huggingface.co/docs/transformers/model_doc/bert, DLA: 22.04.2024

[13] https://fasttext.cc/, DLA: 22.04.2024

[14] https://studieswithcode.com/method/elmo, DLA: 22.04.2024

4.1 Summary of Findings from the Classification

Below are the findings from analyzing trends across all categories.

F1 - Difficulties in Establishing Comparable Assessment Methods. Each study has evaluated its results based on the final goal, making it difficult to compare different approaches. Incorporating a universal evaluation framework alongside specific assessments would be advantageous.

F2 - Absence of Collaboration Across Domains. Papers from different categories do not cite each other, suggesting a lack of interdisciplinary engagement (e.g., approaches extracting LTL automatically can be used in those that focus on generating DECLARE models).

F3 - Increased Interest in Human-Robot Interaction. Recent studies have focused on automatically formalizing temporal requirements involving human conversations, posing the challenge of integrating technical text elements, such as centrifuge times, with human speech. For the latter, a particularly challenging case arises when unrelated information is provided which must be distinguished from actual relevant information.

4.2 Findings from the Input Formats

The input formats used in the included papers are categorized into three distinct types as outlined in [31]: ambiguous (containing vague and unclear expressions), indirect (requiring contextual knowledge for interpretation), and clear (directly leading to a formal specification). This is shown in Fig. 1. Black boxes represent input formats from studies in the augmentation category. White boxes are used for studies related to formal specification generation. Gray boxes mean studies focusing on the automatic generation of modeling notations. A process description, for example, is considered to be indirect since interpreting it requires additional contextual knowledge. Below are all the findings from the analysis of the input formats.

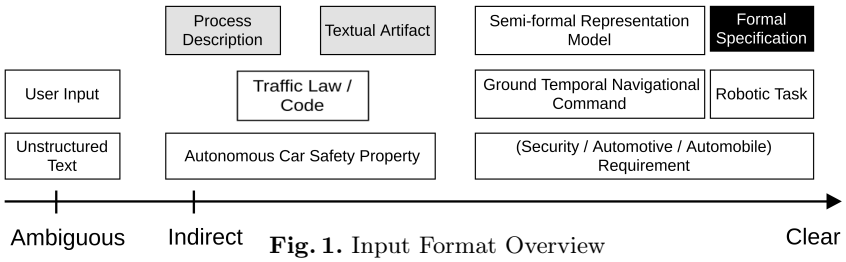

Fig. 1. Input Format Overview

F4 - Inconsistent Terminology. For example, the term *unstructured text* is used variably across studies (e.g., [19], [26], [30]), applying to both brief instructions and entire documents. Terminology should be more aligned and consistent

to avoid confusion and ambiguities in the discussion. Inconsistent terminology mainly hampers comparing findings across different works.

F5 - Insufficient Input Quality. The quality of the input text, e.g., completeness, consistency, relevance, etc., can vary significantly and is not measured. By knowing the input quality, one can optimize further pre-processing tasks, as it is transforming unstructured text into structured requirements.

F6 - Uniform Temporal Information Treatment. In most of the approaches, there was no distinction between functional and non-functional temporal information, a concept introduced in [22]. This can prevent *smells* in temporal specifications, leading to better understanding, easier maintenance, and fewer errors.

F7 - Under-representation of Legal Text. Only five papers specifically considered legal text, indicating a significant gap in integrating legal and regulatory frameworks. Three of them involved manual steps due to higher complexity in compassion to, e.g., robot commands.

4.3 Findings from the Output Formats

This section presents the findings from both how these formats are presented to the user and the various output formats generated by the included papers.

F8 - Poor Context Management. On the one hand, users are asked to provide context, but it is rare to see context included in the output. It would be helpful for users to receive information such as assumptions made by the system, parts that have not been formalized, and any ambiguities found.

F9 - Low Support for Formalizing Quantitative Temporal Information to Formal Specifications. Automatically generated formal specifications like regular expressions, FOL, LTL, and CTL fall short of adequately representing quantitative temporal aspects. In contrast, STL and PPTL offer some capabilities in this area. STL is particularly adept at defining time intervals and quantitative boundaries, making it ideal for scenarios demanding exact timing (e.g., *a car's speed must not exceed 100 km/h within the first 10 seconds after ignition*). PPTL, while proficient in managing both finite and infinite time intervals (e.g., *activity A must occur continuously*), is more focused on qualitative temporal relationships rather than quantitative details. While the RCM discussed in the SLR could address quantitative aspects, it results in a loss of expressiveness. MTL and TCTL are the automatically generated formal specifications that can effectively formalize quantitative and qualitative temporal elements.

F10 - Low Support for Formalizing Quantitative Temporal Information to Modeling Notations. As shown in Tab. 5, for the automatic generation of modeling notations, only the MP-Declare models [1] succeed in capturing quantitative temporal aspects. The Petri Nets, decision models, or DECLARE models would need a time extension to achieve this. Furthermore, standard business process models, as detailed in the survey [12], also do not fully capture quantitative aspects. To effectively handle these aspects, they require a time-extended Business Process Model as, e.g., proposed in [46].

5 NL2MTL Approach and Evaluation

This section introduces a prototype to address the most relevant gaps identified in findings F1, F7, F8, F9, and F10. These include the difficulties in establishing a comparable assessment method, under-representation of legal texts (e.g., useful for real-time compliance in self-driving vehicles), poor output context management (improving usability), and the necessity to automate the formalization of requirements, considering both quantitative and qualitative aspects of time. This addresses RQ3. An overview of the NL2MTL approach is depicted in Fig. 2, and all material, including the implementation, dataset, and (reproducible) evaluation, is available at NL2MTL[3].

5.1 NL2MTL Foundations

The choice of MTL as a formal specification is motivated by its widespread use in system verification and its recent application in legal text formalization. Illustrative examples of MTL application in legal contexts include [61], where MTL is utilized to interpret trigger conditions and logical judgments within Chinese traffic regulations; [34] where MTL is applied to formalize marine traffic rules; and [40] where MTL is used to formalize traffic rules for autonomous vehicles on German interstates, based on the German Road Traffic Regulation. Since LLMs have been successfully utilized for extracting and formalizing LTL and STL formulas, we integrate LLMs, i.e., GPT-4, in our NL2MTL prototype.

Preliminary Steps. We explored the possibility of expanding upon an existing open-source tool. The candidates emerged from the SLR and constitute NL2LTL [26], Lang2LTL, nl2tl [15] and nl2spec [19]. Among those NL2LTL [26], and Lang2LTL [37] are excluded because they formalize LTL. Focusing on tools that additionally extract STL, a comparison between nl2tl [15] and nl2spec [19] revealed that nl2spec is easier to extend due to its modular structure and more comprehensive frontend. Consequently, the development of NL2MTL was pursued by extending nl2spec. Nonetheless, despite nl2spec's emphasis on aiding users in resolving ambiguities, it was not intuitive to interpret the output messages. Specifically, nl2spec did not indicate which parts of the input text were formalized and which were not. It could also not process long documents like legal texts or process descriptions. This led to the development of our own prototype.

Approach. Figure 2 depicts the three main parts of the NL2MTL approach, i.e., accepted input formats, prompt content, and output example for a system specification, which is provided in both JSON and HTML format. This facilitates the user interpretation. It was developed in Python 3.9 and integrates other LLMs as needed. We utilized GPT-4 because it was already retrieving stable and correct results for extracting formal specifications (i.e., when testing the nl2spec framework). The tested input (c.f., Sect. 5.2) includes system specifications, legal texts, and business process descriptions. Each temporal requirement is represented in the output as an MTL formula based on **atomic propositions**. Each proposition contains a description of it, together with the **temporal granularity** (e.g., seconds), identified **ambiguities** (i.e., unclear or vague aspects),

or those **lacking context** (e.g., the temporal adverb *soon* brings uncertainty), and **assumptions** that are made to come up with the final MTL formula. In addition to this, the output includes a **sequence of dependencies** between MTL formulas (e.g., the temporal adverb *soon* might be considered as *in less than 5 minutes*). The output also contains the text that was **not formalized** and explanations for its exclusion.

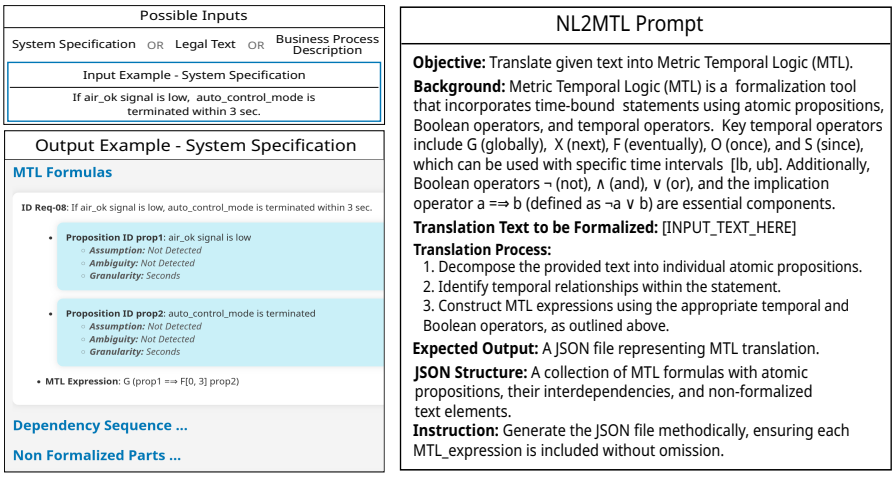

Fig. 2. Overview and Example of NL2MTL Approach

Prompt Design. The structure of the prompt is presented in Fig. 5.1, and the full version is accessible at NL2MTL[3]. It was crafted using the reflection and recipe patterns from the prompt engineering pattern catalog presented in [59]. Adhering to the recipe pattern streamlined the reasoning process for better decision-making. This was achieved by breaking tasks into distinct steps and omitting unnecessary information. The steps involve decomposing the input into individual atomic propositions, elucidating temporal relationships within the statement, and extracting MTL formulas. This structured approach ensures a comprehensive and systematic input analysis, leading to more accurate and contextually relevant outcomes. The reflection pattern enhances ambiguity detection and emphasizes a collaborative approach to information sharing, where both the user and the LLM play integral roles in enriching the context. In the prompt, the semantics of MTL are defined, specifying the temporal and boolean operators that should be considered. In the last part of the prompt, it is indicated the exact expected JSON output structure, each field comes together with its explanation (e.g., for the field *reason_for_non_formalization* it comes along *the reason for the inability to formalize*).

5.2 NL2MTL Evaluation

This section details on the dataset used to evaluate the NL2MTL prototype, the evaluation methodology applied, and the results. The NL2MTL approach

extracts a set of MTL expressions from natural language text for each input format, as demonstrated in Figure 2. To test the stability of the results, this is run five times per input format.

Dataset and Methodology. Table 7 contains an overview of the input files considered to evaluate the NL2MTL prototype, together with the average number per output file of atomic propositions (i.e., *Atom.*), MTL expressions (i.e., *MTL*), assumptions (i.e., *Assu.*), and ambiguities (i.e., *Ambi.*).

Table 7. Dataset Overview and Output Total Averages form Evaluation Run

ID	Description	Atom.	MTL	Assu.	Ambi.
article_78	Article 78 from a Traffic Legal Text [61]	6.75	5.25	4.00	0.50
article_80	Article 80 from a Traffic Legal Text [61]	4.25	2.25	3.00	0.50
syst_req	8 Requirements from CARA Infusion Pump System [47]	12.5	6.25	0.00	0.00
proc_desc	Harvesting Process Description [4]	7.75	6.50	3.25	1.00

The evaluation of the output is based on the four key principles of *good formalization*, as outlined in the methodologies for legal formalization [45]. These include being **correct** (i.e., detecting all atomic propositions and formalizing them following MTL semantics), **transparent** (i.e., defining all used elements, and indicating the reasoning behind a formal specification), **comprehensible** (i.e., easy to interpret), and supportive of **multiple interpretations**.

Results. Table 7 presents the averages of assumptions and ambiguities detected per output file, varying with each round and showcasing NL2MTL's ability to support **multiple interpretations**, all deemed reasonable. This variety raises questions about the practicality and utility of these interpretations for users. All extracted atomic propositions and MTL formulas were **comprehensible** to users. A primary issue noted was in the process descriptions, particularly how time is represented in MTL formulas (e.g., *[0min,30min] vs [30min] vs [0,0.5]*). Symbols and parameters were clearly defined, enhancing **transparency** and preventing hallucination. In cases of unspecified temporal granularity, the system assumes a default setting, which is documented in the assumptions field for user verification. Assessing the **correctness** of the NL2MTL outputs proved challenging. The aim is to confirm that all atomic propositions are correctly represented in the outputs and formalized according to the specified MTL semantics. Errors often arose from parts of text that were not formalized, rather than from the ambiguities or assumptions. In 25% of tests, the NL2MTL system stopped translating subsequent atomic propositions after misinterpreting a stop command in the input text.

6 Discussion

Mitigating Threats to Validity. When conducting an SLR, there is always the threat of missing out on important work. We tried to mitigate this in the following ways. In the first phase of the SLR, the search string was broad to expand the scope, and various digital databases were used to ensure that no studies were overlooked due to publication rights. The inclusion and exclusion

criteria were clearly defined to align the author's perspectives and ensure reproducibility. Additionally, iterative snowballing was employed to identify relevant papers from slightly different fields where our keywords were not used. In *Phase 1*, we identified five surveys related to ours. These surveys were not included in our paper analysis but were instrumental in proving that existing research had not fully covered our research questions. From them, only three surveys contained a comparison of tools designed to automate the process of formalizing temporal requirements. One paper focused solely on extracting LTL [8], while another included various formal specifications [10]. Similarly, [11] involved converting natural language into LTLf formulas for workflow construction. Neither survey rigorously demonstrated the criteria used to include certain studies over others. Additionally, two other surveys were centered on the analysis of automatic requirement formalization [55, 56], which scarcely included methods for formalizing temporal aspects.

Limitations. The NL2MTL approach addressed five elicited findings. In the following, we highlight how the remaining ones could be addressed in future work. For F2, one approach could be to analyze papers excluded by EC4 and EC5, and compare their methodologies with those described in our survey. This could also contribute to addressing F3, as among the cross-domain papers, there is a keen research interest in improving the automatic formalization of natural language for robot-human communication. On the other hand, to adequately cover F4 and F5, concentrating on a specific domain, such as robotics or self-driving vehicles, would be advantageous because these areas have a more specialized vocabulary. Similarly, when working on F6, it would be simpler to first distinguish between functional and non-functional temporal information within a specific domain.

7 Conclusion

This paper features a systematic literature review to address how to effectively compare approaches aiming at extracting and formalizing temporal requirements from text. We classified approaches along their goal and output format. In total, the literature analysis revealed ten findings. To address five of those, we developed NL2MTL which closes a significant gap, i.e., existing approaches mainly focus on qualitative temporal requirements (e.g., LTL). However, quantitative aspects, such as representing exact time units (e.g., STL), are equally important. NL2MTL allows for an automatic translation of system specifications, legal texts, and business process descriptions to MTL utilizing the power of state-of-the-art LLMs. The evaluation of NL2MTL along the key principles of a *good formalization* shows promising results regarding correctness, comprehensiveness, transparency, and interpretation of the results for all input types. Future work can focus on testing the prototype in a real-world scenario and conducting in-depth user studies with domain experts on its usefulness.

Acknowledgements: This work has been funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) –project number 277991500.

References

1. van der Aa, H., Balder, K.J., Maggi, F.M., Nolte, A.: Say it in your own words: Defining declarative process models using speech recognition (2020). https://doi.org/10.1007/978-3-030-58638-6_4
2. van der Aa, H., Ciccio, C.D., Leopold, H., Reijers, H.A.: Extracting declarative process models from natural language (2019). https://doi.org/10.1007/978-3-030-21290-2_23
3. Alman, A., Balder, K.J., Maggi, F.M., van der Aa, H.: Declo: A chatbot for user-friendly specification of declarative process models (2020), https://ceur-ws.org/Vol-2673/paperDR12.pdf
4. Barrientos, M., Winter, K., Mangler, J., Rinderle-Ma, S.: Verification of quantitative temporal compliance requirements in process descriptions over event logs (2023). https://doi.org/10.1007/978-3-031-34560-9_25
5. Bellan, P., Dragoni, M., Ghidini, C.: Extracting business process entities and relations from text using pre-trained language models and in-context learning (2022). https://doi.org/10.1007/978-3-031-17604-3_11
6. Bertram, V., Kausch, H., Kusmenko, E., Nqiri, H., Rumpe, B., Venhoff, C.: Leveraging natural language processing for a consistency checking toolchain of automotive requirements (2023). https://doi.org/10.1109/RE57278.2023.00029
7. Bombieri, M., Meli, D., Dall'Alba, D., Rospocher, M., Fiorini, P.: Mapping natural language procedures descriptions to linear temporal logic templates: an application in the surgical robotic domain (2023). https://doi.org/10.1007/s10489-023-04882-0
8. Brunello, A., Montanari, A., Reynolds, M.: Synthesis of LTL formulas from natural language texts: State of the art and research directions (2019). https://doi.org/10.4230/LIPIcs.TIME.2019.17
9. Burattin, A., Maggi, F.M., Sperduti, A.: Conformance checking based on multi-perspective declarative process models (2016). https://doi.org/10.1016/J.ESWA.2016.08.040
10. Buzhinsky, I.: Formalization of natural language requirements into temporal logics: a survey (2019). https://doi.org/10.1109/INDIN41052.2019.8972130
11. Chakraborti, T., Rizk, Y., Isahagian, V., Aksar, B., Fuggitti, F.: From natural language to workflows: Towards emergent intelligence in robotic process automation (2022). https://doi.org/10.1007/978-3-031-16168-1_8
12. Cheikhrouhou, S., Kallel, S., Guermouche, N., Jmaiel, M.: The temporal perspective in business process modeling: a survey and research challenges (2015). https://doi.org/10.1007/S11761-014-0170-X
13. Chen, B., Chen, K., Hassani, S., Yang, Y., Amyot, D., Lessard, L., Mussbacher, G., Sabetzadeh, M., Varró, D.: On the use of GPT-4 for creating goal models: An exploratory study (2023). https://doi.org/10.1109/REW57809.2023.00052
14. Chen, Y., Gandhi, R., Zhang, Y., Fan, C.: NL2TL: transforming natural languages to temporal logics using large language models (2023), https://aclanthology.org/2023.emnlp-main.985
15. Chen, Z., Chen, W., Zha, H., Zhou, X., Zhang, Y., Sundaresan, S., Wang, W.Y.: Logic2text: High-fidelity natural language generation from logical forms (2020). https://doi.org/10.18653/V1/2020.FINDINGS-EMNLP.190
16. Chen, Z., Li, I., Zhang, H., Preum, S.M., Stankovic, J.A., Ma, M.: Cityspec: An intelligent assistant system for requirement specification in smart cities (2022). https://doi.org/10.1109/SMARTCOMP55677.2022.00020

17. Chhabra, A., Sangroya, A., Anantaram, C.: Formalizing and verifying natural language system requirements using petri nets and context based reasoning (2018), https://ceur-ws.org/Vol-2134/paper09.pdf
18. Conrad, E., Titolo, L., Giannakopoulou, D., Pressburger, T., Dutle, A.: A compositional proof framework for fretish requirements (2022). https://doi.org/10.1145/3497775.3503685
19. Cosler, M., Hahn, C., Mendoza, D., Schmitt, F., Trippel, C.: nl2spec: Interactively translating unstructured natural language to temporal logics with large language models (2023). https://doi.org/10.1007/978-3-031-37703-7_18
20. Costescu, D.M.: Building on a traffic code violating monitor for autonomous vehicles: Trio overtaking model (2020)
21. Donadello, I., Riva, F., Maggi, F.M., Shikhizada, A.: Declare4py: A python library for declarative process mining (2022), https://ceur-ws.org/Vol-3216/paper_249.pdf
22. Eder, J., Franceschetti, M., Lubas, J.: Time and processes: Towards engineering temporal requirements (2021). https://doi.org/10.5220/0010625400090016
23. Etikala, V., Veldhoven, Z.V., Vanthienen, J.: Text2dec: Extracting decision dependencies from natural language text for automated DMN decision modelling (2020). https://doi.org/10.1007/978-3-030-66498-5_27
24. Fill, H., Fettke, P., Köpke, J.: Conceptual Modeling and Large Language Models: Impressions From First Experiments With ChatGPT (2023). https://doi.org/10.18417/emisa.18.3
25. Fontenla-Seco, Y., Winkler, S., Gianola, A., Montali, M., Penín, M.L., Diz, A.J.B.: The droid you're looking for: C-4pm, a conversational agent for declarative process mining (2023), https://ceur-ws.org/Vol-3469/paper-20.pdf
26. Fuggitti, F., Chakraborti, T.: NL2LTL - a python package for converting natural language (NL) instructions to linear temporal logic (LTL) formulas (2023). https://doi.org/10.1609/aaai.v37i13.27068
27. Gavran, I., Darulova, E., Majumdar, R.: Interactive synthesis of temporal specifications from examples and natural language (2020). https://doi.org/10.1145/3428269
28. Ge, N., Yang, J., Yu, T., Liu, W.: AutoMTLSpec: Learning to Generate MTL Specifications from Natural Language Contracts (2023). https://doi.org/10.1109/ICECCS59891.2023.00018
29. Golra, F.R., Dagnat, F., Souquières, J., Sayar, I., Guérin, S.: Bridging the gap between informal requirements and formal specifications using model federation (2018). https://doi.org/10.1007/978-3-319-92970-5_4
30. Hahn, C., Schmitt, F., Tillman, J.J., Metzger, N., Siber, J., Finkbeiner, B.: Formal specifications from natural language (2022). https://doi.org/10.48550/ARXIV.2206.01962
31. He, J., Bartocci, E., Nickovic, D., Isakovic, H., Grosu, R.: Deepstl - from english requirements to signal temporal logic (2022). https://doi.org/10.1145/3510003.3510171
32. Ivanov, V., Sadovykh, A., Naumchev, A., Bagnato, A., Yakovlev, K.: Extracting software requirements from unstructured documents (2022), https://arxiv.org/abs/2202.02135
33. Kitchenham, B.: Procedures for performing systematic reviews (2004)
34. Krasowski, H., Althoff, M.: Temporal logic formalization of marine traffic rules (2021). https://doi.org/10.1109/IV48863.2021.9575685

35. Li, C., Chang, J., Wang, X., Zhao, L., Mao, W.: Formalization of natural language into PPTL specification via neural machine translation (2022). https://doi.org/10.1007/978-3-031-29476-1_7
36. Lin, J., Zhou, W., Wang, H., Cao, Z., Yu, W., Zhao, C., Zhao, D., Yang, D., Li, J.: Road traffic law adaptive decision-making for self-driving vehicles (2022). https://doi.org/10.1109/ITSC55140.2022.9922208
37. Liu, J.X., Yang, Z., Idrees, I., Liang, S., Schornstein, B., Tellex, S., Shah, A.: Grounding complex natural language commands for temporal tasks in unseen environments (2023)
38. López, H.A., Marquard, M., Muttenthaler, L., Strømsted, R.: Assisted declarative process creation from natural language descriptions (2019). https://doi.org/10.1109/EDOCW.2019.00027
39. López, H.A., Strømsted, R., Niyodusenga, J., Marquard, M.: Declarative process discovery: Linking process and textual views (2021). https://doi.org/10.1007/978-3-030-79108-7_13
40. Maierhofer, S., Rettinger, A., Mayer, E.C., Althoff, M.: Formalization of interstate traffic rules in temporal logic (2020). https://doi.org/10.1109/IV47402.2020.9304549
41. Manas, K., Paschke, A.: Semantic Role Assisted Natural Language Rule Formalization for Intelligent Vehicle (2023). https://doi.org/10.1007/978-3-031-45072-3_13
42. Mustroph, H., Barrientos, M., Winter, K., Rinderle-Ma, S.: Verifying resource compliance requirements from natural language text over event logs (2023). https://doi.org/10.1007/978-3-031-41620-0_15
43. Nayak, A., Timmapathini, H., Murali, V., Ponnalagu, K., Venkoparao, V.G., Post, A.: Req2spec: Transforming software requirements into formal specifications using natural language processing (2022). https://doi.org/10.1007/978-3-030-98464-9_8
44. Neuberger, J., Ackermann, L., Jablonski, S.: Beyond rule-based named entity recognition and relation extraction for process model generation from natural language text (2023). https://doi.org/10.48550/arXiv.2305.03960
45. Novotná, T., Libal, T.: An evaluation of methodologies for legal formalization (2022). https://doi.org/10.1007/978-3-031-15565-9_12
46. Ocampo-Pineda, M., Posenato, R., Zerbato, F.: Timeawarebpmn-js: An editor and temporal verification tool for time-aware BPMN processes (2022). https://doi.org/10.1016/J.SOFTX.2021.100939
47. Osama, M., Zaki-Ismail, A., Abdelrazek, M.A., Grundy, J., Ibrahim, A.S.: A comprehensive requirement capturing model enabling the automated formalisation of NL requirements (2023). https://doi.org/10.1007/s42979-022-01449-7
48. Pan, J., Chou, G., Berenson, D.: Data-efficient learning of natural language to linear temporal logic translators for robot task specification (2023). https://doi.org/10.1109/ICRA48891.2023.10161125
49. Pi, X., Shi, J., Huang, Y., Wei, H.: Automated Mining and Checking of Formal Properties in Natural Language Requirements (2019). https://doi.org/10.1007/978-3-030-29563-9_8
50. Quartey, B., Rosen, E., Tellex, S., Konidaris, G.: Verifiably following complex robot instructions with foundation models (2024). https://doi.org/10.48550/ARXIV.2402.11498
51. Quishpi, L., Carmona, J., Padró, L.: Extracting decision models from textual descriptions of processes (2021). https://doi.org/10.1007/978-3-030-85469-0_8

52. Sànchez-Ferreres, J., Burattin, A., Carmona, J., Montali, M., Padró, L.: Formal reasoning on natural language descriptions of processes (2019). https://doi.org/10.1007/978-3-030-26619-6_8
53. Sànchez-Ferreres, J., Burattin, A., Carmona, J., Montali, M., Padró, L., Quishpi, L.: Unleashing textual descriptions of business processes (2021). https://doi.org/10.1007/s10270-021-00886-x
54. Sharma, S., Brian Lee, K.M., Brown, M., Best, G.: Instructing Robots with Natural Language via Bi-RNNs for Temporal Logic Translation (2023), https://www.scopus.com/inward/record.uri?eid=2-s2.0-85184383465&partnerID=40&md5=9a09f07a3d2022b763a0a17f7d14289d
55. Sonbol, R., Rebdawi, G., Ghneim, N.: The use of nlp-based text representation techniques to support requirement engineering tasks: A systematic mapping review (2022). https://doi.org/10.1109/ACCESS.2022.3182372
56. Sudhi, V., Kutty, L., Gröpler, R.: Natural language processing for requirements formalization: How to derive new approaches? (2023). https://doi.org/10.48550/arXiv.2309.13272
57. Wang, C., Ross, C., Kuo, Y.L., Katz, B., Barbu, A.: Learning a natural-language to LTL executable semantic parser for grounded robotics (2020), https://www.scopus.com/inward/record.uri?eid=2-s2.0-85168241969&partnerID=40&md5=6c3e5cd9fe6da29032fae93808c78a09
58. Wang, X., Li, G., Li, C., Zhao, L., Shu, X.: Automatic generation of specification from natural language based on temporal logic (2020). https://doi.org/10.1007/978-3-030-77474-5_11
59. White, J., Fu, Q., Hays, S., Sandborn, M., Olea, C., Gilbert, H., Elnashar, A., Spencer-Smith, J., Schmidt, D.C.: A prompt pattern catalog to enhance prompt engineering with chatgpt (2023). https://doi.org/10.48550/ARXIV.2302.11382
60. Wu, X., Cai, Y., Lian, Z., Leung, H., Wang, T.: Generating natural language from logic expressions with structural representation (2023). https://doi.org/10.1109/TASLP.2023.3263784
61. Yu, W., Zhao, C., Wang, H., Liu, J., Ma, X., Yang, Y., Li, J., Wang, W., Hu, X., Zhao, D.: Online Legal Driving Behavior Monitoring for Self-driving Vehicles (2024). https://doi.org/10.6084/m9.figshare.24372535.v1
62. Zaki-Ismail, A., Osama, M., Abdelrazek, M., Grundy, J., Ibrahim, A.S.: RCM: requirement capturing model for automated requirements formalisation (2021). https://doi.org/10.5220/0010270401100121
63. Zaki-Ismail, A., Osama, M., Abdelrazek, M., Grundy, J.C., Ibrahim, A.S.: ARF: automatic requirements formalisation tool (2021). https://doi.org/10.1109/RE51729.2021.00060
64. Zaki-Ismail, A., Osama, M., Abdelrazek, M.A., Grundy, J.C., Ibrahim, A.S.: Rcm-extractor: an automated nlp-based approach for extracting a semi formal representation model from natural language requirements (2022). https://doi.org/10.1007/s10515-021-00312-y
65. Zhang, S., Zhai, J., Bu, L., Chen, M., Wang, L., Li, X.: Automated Generation of LTL Specifications for Smart Home IoT Using Natural Language (2020)
66. Zhao, L., Alhoshan, W., Ferrari, A., Letsholo, K.J., Ajagbe, M.A., Chioasca, E., Batista-Navarro, R.T.: Natural language processing for requirements engineering: A systematic mapping study (2022). https://doi.org/10.1145/3444689