Formalization of natural language requirements into temporal logics: a survey

Igor Buzhinsky

Citation:

Buzhinsky I. Formalization of natural language requirements into temporal logics: a survey. 17th IEEE International Conference on Industrial Informatics (INDIN). July 22–25, 2019, Helsinki-Espoo, Finland, pp. 400–406. IEEE, 2019.

DOI: https://doi.org/10.1109/INDIN41052.2019.8972130

Formalization of natural language requirements into temporal logics: a survey

Igor Buzhinsky
Department of Electrical Engineering and Automation, Aalto University, Finland
Computer Technology Department, ITMO University, St. Petersburg, Russia
igor.buzhinskii@aalto.fi

***Abstract*—One of the challenges of requirements engineering is the fact that requirements are often formulated in natural language. This represents difficulty if requirements must be processed by formal approaches, especially if these approaches are intended to check the requirements. In model checking, a formal technique of verification by exhaustive state space exploration, requirements must be stated in formal languages (most commonly, temporal logics) which are essentially supersets of the Boolean propositional logic. Translation of natural language requirements to these languages is a process which requires much knowledge and expertise in model checking as well the ability to correctly understand these requirements, and hence automating this process is desirable. This paper reviews existing approaches of requirements formalization that are applicable for, or at least can be adapted to generation of discrete time temporal logic requirements. Based on the review, conclusions are made regarding the practical applicability of these approaches for the considered problem.**

I. Introduction

Safety-critical automation systems demand extensive efforts in ensuring their correctness and reliability. One of the means of doing this is model checking [1], [2], a formal verification approach that is based on exhaustive state space exploration. In Finland, for example, model checking is used to assess the correctness of instrumentation and control (I&C) systems of nuclear power plants (NPPs) [3].

Requirements to systems to be model-checked must be written in formal languages, such as temporal logics, examples of which are linear temporal logic (LTL) and computation tree logic (CTL). Converting requirements written in natural language (NL)[1], which is their most common form of representation, to temporal logics is usually done manually. However, requirements formalization is a difficult task [4]: it requires both expertise in formal methods and understanding of the domain. Thus, the formalization process is time-consuming and error-prone. Automating this conversion by applying natural language processing (NLP) techniques would simplify the work of analysts and reduce the probability of error, and, in the long run, may widen the use of model checking in industry, which is currently quite limited.

This study is a survey of existing approaches that are applicable for NL requirements formalization into discrete time temporal logics or can be potentially adapted to solve this problem. Two classes of approaches are examined. The first class is *direct approaches*, which often employ formal language theory: for example, the translation process may be based on parsing an NL requirement according to a context-free grammar, and then generating a temporal representation based on the parse tree. The second class is *statistical machine translation approaches* [5]: although their intention is to translate NLs between each other, the requirements formalization process can be treated as language translation. This classification roughly corresponds to *rule-based* and *statistical approaches* in NLP. The applicability of the approaches of both types to the considered problem is assessed.

The paper is organized as follows. In Section II, the concepts used throughout it are introduced. In Sections III and IV, direct requirements formalization approaches and machine translation approaches are considered, respectively. In Section V, related work is briefly mentioned. Finally, in Section VI, the results of the survey are analyzed, and conclusions are made.

II. Background

A. Model checking

Model checking is a formal verification technique that is based on exhaustive state space exploration. Its main area of application is verification of mission-critical and safety-critical systems. The system to be verified is represented as a *formal model* using the language of a *model checker*, a tool that implements model checking algorithms. Such a representation may be graphical [6] or textual [7], [8]. The requirements to be verified are also represented in formal languages, such as *linear temporal logic* (LTL), *computation tree logic* (CTL), or property specification language (PSL). The most popular formal languages are textual, although graphical languages exist as well [9].

In particular, LTL and CTL are extensions of the Boolean propositional logic. While the instantaneous state of the verified system can be described with propositional Boolean formulas (such as $(a = 5) \wedge \neg b$, where a is some integer variable and b is some Boolean variable), entire behavior traces of the system require a notion of time. In LTL, time is modeled using a logical clock, and *temporal operators* specify the requirements for multiple time instants (which can be called steps or cycles). For example, temporal operators $\mathbf{G}$ ("always"), $\mathbf{F}$ ("in the future") and $\mathbf{X}$ ("next") state that their

[1]NL is a generic term referring to any language, though the majority of the works are dedicated to English.

arguments are satisfied for all future time instants, for some future time instant, and for the next time instant, respectively.

While checked traces in LTL are always linear, CTL utilizes a branching semantics: each state of the formal model is a source of multiple possible traces. Correspondingly, CTL operators are prefixed with *path quantifiers* $\mathbf{A}$ ("for all paths") and $\mathbf{E}$ ("for some path"). For example, the formula $\mathbf{EF}\,x$ means that x is reachable from the current state of the model.

B. Formal languages and grammars

A *formal language* [10] is a set of finite sequences of elements of a finite set Σ called an *alphabet*. A *formal grammar* is a particular way of specifying a formal language using *production rules*. Informally, these rules are string replacements, and each possible word of the grammar can be obtained as a result of a finite number of such replacements applied to the designated *start symbol* S. Production rules are defined over a larger alphabet $\Sigma \cup N$, where N is the set of *nonterminals*, and $S \in N$.

In *context-free grammars*, each nonterminal can be treated as a language construct that can be recursively expressed using other constructs (elements of N) and symbols (elements of Σ). This leads to the possibility of explaining the derivation of each word of the grammar as a *parse tree*, whose leaves comprise the derived word and whose internal nodes correspond to applied production rules. For example, if one needs to define a context-free grammar for LTL, this grammar may include a nonterminal corresponding to a formula, and the rules explaining how a formula can be obtained: it is either a variable, a constant, or a Boolean or a temporal operator applied to one or two formulas. Thus, for each LTL formula, a parse tree can be constructed with variables and constants as leaves and operators as internal nodes.

In practice, recognition of context-free grammars is done by software tools called *parsers*. If the supplied word belongs to the language of the grammar, the parser will construct a parse tree during the recognition process. Parsers can be also used as a means of translation between similar grammars.

C. Translation of natural languages requirements to temporal logics

In this paper, we will use requirements from the domain of nuclear I&C systems as examples. To begin with, consider a simple NL requirement:

- R1: *If the pressurizer water level rises above l_0, then the reactor shall be tripped* (i.e., shut down) *on the next cycle at latest.*

In this requirement, l_0 is a constant (say, 1.1 m), but we will use the symbolic name l_0 to avoid the discussion of real value discretization. Similarly, the words "on the next cycle" may come from a concrete maximum response time (e.g., 250 ms). A reasonable formalization of R1 into LTL looks as follows:

$$f_{\text{R1}} = \mathbf{G}((l > l_0) \rightarrow (t \vee \mathbf{X}\,t)).$$

The overall grammatical structure of f_{R1} is close to the one of R1: the *"if ..., then"* statement is formalized as implication ($\rightarrow$). The translation of words *"the reactor shall be tripped on the next cycle at latest"* is less direct: assuming that t is the reactor trip signal, without the words *"at latest"*, this part of the requirement would be translated as $\mathbf{X}\,t$, but the addition of the words *"at latest"* also allows the signal to be generated already at the current cycle: $t \vee \mathbf{X}\,t$. An even larger distinction is the presence of the outermost $\mathbf{G}$ operator: R1 does not explicitly state that the entire requirement must be satisfied on each step of the behavior trace, but this assumption is *implied*.

Below, a more difficult example is given:

- R2: *If steam pressures in either of the steam generators exceed the mean pressure by Δp and emergency feedwater lines are not closed, then the corresponding valve closing signals shall be generated.*

To begin with, R2 contains a phrase *"steam pressures in either of the steam generators"* that needs to be translated into a disjunction of variables, one for each steam generator. The number of steam generators, however, is not given in R2. Similarly, the phrase *"the mean pressure"* should be transformed into an arithmetic formula whose contents must be guessed from the beginning of R2.

A somewhat larger difficulty arises in the second part of R2. Do *"the corresponding valve closing signals"* refer to all the signals, or only to the ones which correspond to steam generators whose pressures exceed the mean pressure? Although the second understanding may appear more logical, it cannot be inferred from the syntactic structure of R2. This is an example of the *ambiguity problem* of NLs, which makes automatic NL understanding especially challenging. A second instance of this problem comes from the use of the word *"shall"*: it is not clear whether the signals must be generated immediately (no temporal operator) or eventually (temporal operator $\mathbf{F}$).

One reasonable translation of R2 to LTL, assuming that the number of steam generators is six, may look as follows:

$$f_{\text{R2}} = \bigwedge_{i=1}^{6} \mathbf{G}((p_i - \Delta p \geq (p_1 + ... + p_6)/6) \rightarrow \mathbf{F}\,c_i).$$

Here, the outermost conjunction has been used to simplify the representation of f_{R2}; the actual LTL representation would involve six explicit occurrences of the $\mathbf{G}$ statement. Although an extension of the LTL syntax may be considered to avoid such duplication, the iteration over steam generators seems difficult to guess automatically.

To sum up, the problem of requirements translation to temporal logics contains the following main challenges:

1) ambiguity of NL, including the ambiguity of particular words (such as pronouns) and the ambiguity of the meaning of the entire requirements;
2) changes in the grammatical structure of the NL requirement when translating it to a temporal logic (such as repeating certain subformulas multiple times).

Below, we review attempts to address this problem with the focus on temporal logics as target languages in the

translation task. Often, a *parallel corpus* is available as initial data: this is a set of pairs of an NL requirement and its corresponding formal version. An approach utilizing a parallel corpus may explicitly incorporate the knowledge of this corpus (for example, by manually constructing two formal grammars and specifying the translation procedure between them), or by using it as input data to an automated model construction approach (for example, by using statistical or neural machine translation).

III. Direct formalization approaches

The approaches to requirements formalization reviewed in this section are based on language modeling with discrete objects, such as formal grammars. We start with basic approaches and proceed with more elaborate ones.

A. Pattern-based approaches

Probably the simplest solution to the formalization problem is to establish direct correspondence between certain types of temporal formulas and their possible NL representations. Such an approach is used in [11], utilizing the earlier idea of *patterns* of temporal properties [12], [13]. According to [12], a pattern is a "generalized description of a commonly occurring requirement on the permissible state/event sequences in a finite-state model of a system." For example, the "response" pattern establishes a causal relationship between two events and is represented by LTL formulas of the form $\mathbf{G}(x \to \mathbf{F}\, y)$, where x and y are ordinary Boolean formulas. Thus, f_{R2} is a conjunction of six instances of this pattern. In [14], [15], structured English grammars are presented that enable automatic conversion of limited subsets of English to the patterns introduced in [12], [13]. Then, in [11], the grammar from [15] is integrated into a tool suite for specification and analysis of requirements to UML models, which supports adaptation to the writing style and vocabulary used in requirements for a particular domain. Requirement templates which allow formalization by substituting placeholders with concrete statements are also known as *boilerplates* [16].

While the use of patterns and boilerplates offers a solution to convert NL requirements to temporal logics, they only solve this problem for requirements prepared according to specific guidelines. This reduces the practical applicability of this solution, although formulating requirements accounting for subsequent formal verification is a practice that makes errors during requirements formalization much less probable.

B. Approaches based on fixed translation models

In the work [17], the correspondence between NL and the temporal logic ACTL, an extension of CTL with actions, is established by means of a formal grammar. Differently from [15], the work [17] supports formulation of arbitrary formulas of the considered temporal logic instead of a fixed set of patterns. This grammar was developed based on a corpus of NL expressions and thus supports some flexibility in formulation of the requirements. This corpus included examples with ambiguous meanings, for which multiple translations to ACTL were possible. Detection of these ambiguities was incorporated into the translation approach: when an input sentence is recognized as ambiguous, the missing information is asked from the user. On one hand, this makes the entire solution of the translation problem partial as it is not fully automated; but on the other hand, the resultant temporal requirements must be correct, and hence generation of formal requirements when an ambiguity is detected is unreliable. In addition to resolving such ambiguities, the user must specify the words used in their domain (e.g., "pressurized water level" in the case of R1).

In [18], a different approach to transform NL requirements is proposed that is grounded on discourse representation theory [19]. According to the authors, this makes the supported fragment of NL much more flexible than in [17] since it is not explicitly bound to the structure of the temporal logic. In addition, unlike in [17], NL requirements may span through multiple sentences and are allowed to use pronouns (the user may be asked to resolve the meaning of pronouns and some other words, though). Confusingly, the target temporal logic in [18] is also ACTL, but this formalism is different from the one used in [17] and is actually a subset of CTL without $\mathbf{E}$ path quantifiers, meaning that this logic is less rich than the one used in [17].

The works [20] and [21] propose solutions specifically for the domain of robot control. In [20], similarly to [17], a strict NL grammar is developed according to which the requirements must be formulated. The differences are the use of a fragment of LTL instead of ACTL as the target formal language and the suitability of this solution specifically for the robot control problem: the task for the robot is described as $\varphi_e \to \varphi_s$, where φ_e and φ_c are requirements for the environment and for the robot, respectively, both restricted to a particular form. In [21], where a different domain-specific grammar is developed, in addition to producing a temporal (CTL*) formula based on an NL requirement, a set of instructions is generated for the robot to achieve this requirement. This additional benefit, unfortunately, is not useful in the context of requirements formalization for subsequent model checking. Moreover, the logic CTL* is computationally difficult to model-check and is not supported by popular model checkers. As for the supported NL requirements, they need to be complete and correctly structured, like in [17].

Similarly to [20], [21], a domain-specific grammar is designed in [22], which is done based on a corpus of NL requirements for code in Hardware Description Language (HDL), a language to describe the behavior and structure of electronic circuits. The approach does not require any input from the user to resolve ambiguities; nevertheless, the authors report a high success rate of automatic translation (91%). As a general approach to resolve ambiguities in NL requirements, manual incorporation of translation rules based on the available corpus is proposed (otherwise, the proposed translation system may produce an incorrect temporal formula).

C. Approach based on grammar learning

The idea of [22] is developed in [23], where the grammar of the NL subset used in the training corpus is *learned automatically*. This was done from a surprisingly small set of 17 requirements; yet, the authors report high quality of translating requirements that were unknown during grammar learning. On the other hand, each requirement was represented as a single sentence, and the variability of the language in all the requirements was very low. The target formalism used in [22] is SystemVerilog assertions [24], which is similar to LTL by considering linear system behavior traces. Nevertheless, there is evidence [25] that such a learning approach may improve the work [22] and thus allow translation to CTL. Moreover, adaptation of this translation approach to other temporal logics may also be possible.

D. Approaches based on processing NL parse trees

So far we have seen that a correspondence between NL and a temporal logic can be established by mirroring the structure of the temporal logic in NL [17] or by constructing a formal grammar for a particular domain [20]–[22]. One more general approach is to initially consider a rather large fragment of NL and by means of NL parsing obtain a *syntax tree* wherein the grammatical roles of words are identified. This approach is taken in [26], [27]. In [26], the mobile robots domain is again considered. Verbs and their arguments are extracted from the syntax tree, transforming it to a set of *frames*. Then, a predefined set of LTL patterns is recognized by associating them to particular types of frames. In [27], the proposed structured NL grammar for LTL is not limited to a particular domain. Atomic propositions are found, after which hand-crafted rules are applied to the syntax tree to extract logical and temporal ($\mathbf{G}$ and $\mathbf{F}$) operators.

The ideas of NL parsing that are applied in [26], [27] are enhanced in [28]. The parse tree (called the *dependency graph*) is semantically processed by identifying arithmetic expressions, unique entities and events, and then applying domain-independent type rules, whose purpose is to transform linguistic patterns (e.g., connections of words with certain prepositions) into predicates with defined semantics. These predicates are then represented as a *predicate graph*, from which an LTL formula is generated by recursively applying translation rules. The approach was evaluated on four corpora of requirements consisting of 300 sentences in total. Once the exact rules of the translation approach were tuned on 112 of them (selected from only two corpora), its performance on the remaining ones was found to be reasonably good.

IV. Requirements formalization by means of machine translation

A. Statistical machine translation

Statistical machine translation [29] is a field that studies translation between different NLs using probabilistic models. The language translation problem is treated as a *supervised machine learning* [30] one: that is, the translation procedure is not explicitly programmed but learned automatically from a parallel corpus. Suppose that each sentence is translated separately, and the task is to learn how to translate sentences f in a source language to sentences e in a target language, given a parallel corpus. The approach usually taken in statistical machine translation is to find the translation e with the maximum probability, which can be expressed using the Bayes rule: $p(e|f) = p(f|e)p(e)/p(f)$. Since $p(f)$ is the same for all possible translations e, the task is to find $\arg\max_e p(f|e)p(e)$. This is done by separately considering the *language model* $p(e)$ and the *translation model* $p(f|e)$.

The language model $p(e)$ assigns probabilities to sequences of words. The simplest language model is based on counting the numbers of all n-grams, or sequences of n words, in the target language part of the corpus. The probability of the entire sequence is calculated using the chain rule, and the probability of each word in the sequence is estimated based on the last $n-1$ words.

In the translation model $p(f|e)$, one can assume that the sentence f is decomposed into I *phrases* $f_1...f_I$ that are separately translated into phrases $e_1...e_I$ of the sentence e. The order of translated phrases can be changed (i.e., $e \neq e_1...e_I$). Under these assumptions, $p(f|e)$ can be expressed with $p(f_1...f_I|e_1...e_I)$, and the latter probabilities, in turn, are products of probabilities of translating each f_i into e_i ($1 \leq i \leq I$) weighted by a distortion (reordering) model. The exact ways to learn phrase translation probabilities and to model distortion may vary.

One of the popular measures of translation quality is BLEU [31], whose idea is to calculate the percentage of exactly matched n-grams between the generated and the reference translation. The work [5] gives BLEU scores for various translation models evaluated on German to English translation. With a corpus of 320 thousand sentence pairs, BLEU scores up to 0.25–0.29 can be reached. With only 10 thousand sentence pairs, the corresponding scores are around 0.20–0.22.

B. Adapting machine translation to requirements formalization

Assuming that sentences e belong to a temporal logic and sentences f are requirements formulated in NL, it is possible to view machine translation as a requirements formalization process. In Fig. 1, a fictitious example of phrase extraction and reordering is provided for a translation of R1 into f_{R1}. Although R1 is represented by a relatively simple NL sentence, and the example is simplified by omitting mappings for parentheses, the mapping of phrases is already quite complicated. Moreover, this mapping is not one-to-one. In particular, the translations of some phrases may interleave with each other, and the temporal operator $\mathbf{G}$ is not mapped to any phrase, although such situations are also possible in translations between NLs.

In NL translation, BLEU scores of 0.25–0.29 (with a theoretical maximum of 1.0) reported in [5] may appear low but they are often sufficient to understand the meaning of the

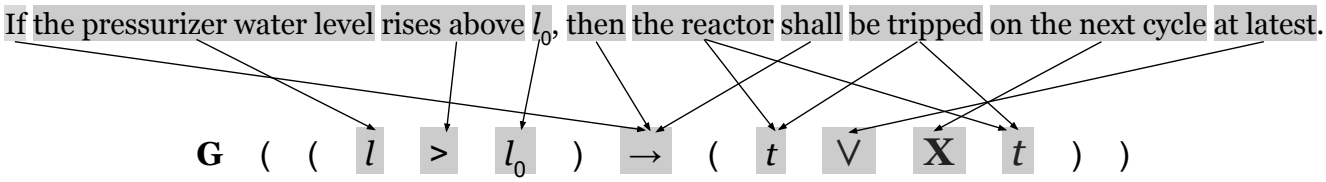

Fig. 1. Example of phrase extraction and reordering when translating R1 into LTL.

translated text. Unfortunately, if the target language is a formal one, a slight change applied to the translation may make it syntactically incorrect or change the meaning entirely. Thus, application of such statistical translation for requirements formalization may only be possible if a human expert examines and corrects the results of automatic translation.

C. Classification of requirement patterns

In general, in supervised machine learning, a model that predicts desired outputs from given inputs is constructed based on *training data* (e.g., a corpus). In Section IV-D, we will see how this idea can be applied to machine translation, but by now we consider a simpler problem: attribute an NL requirement to one of the predefined classes. In [32], these classes are selected as the eight most frequent LTL patterns among the ones proposed by Dwyer [12], [13]. To solve this classification problem, typical NLP procedures are used: stemming, representation of texts as multidimensional vectors based on word counts and tf-idf [33].

D. Neural machine translation

Neural networks [34] are models that are popular in machine learning and are capable of approximating complex nonlinear functions. Their architecture is inspired by brain neurons: a typical neural network is composed of a number of simple processing units arranged in several layers. *Learning* of a neural network is performed by iteratively processing all available training data, usually at least several times, and adjusting the parameters of the network using back-propagation [35] to reduce the prediction error on these data.

Recently, neural networks have become widely used in machine translation [36], [37]. Their competitive advantage over traditional statistical approaches is learning word representations (*embeddings*) as real-valued vectors [38], while traditional statistical approaches operate with n-gram models with a typical $n = 3$, losing much information. The use of $n \leq 3$ is due to the problems of the total number of n-grams and generalization to unseen n-grams. Support of long contexts by neural networks is achieved by reusing the internal state of the network for each new word of the sentence being translated. Models implementing this idea are known as *recurrent neural networks* and are intended to deal with sequential data; their most common architectures are long short-term memory (LSTM) and gated recurrent units (GRU).

Although neural networks can outperform traditional statistical approaches, they need large training corpora to do so. According to a recent experimental study [39], where a translation from English to Spanish was evaluated, neural machine translation starts outperforming phrase-based statistical translation at about 15 million English words in the training corpus (with a BLEU score of 25.7). With 385.7 million words, neural translation reaches a BLEU score of 31.1. Unfortunately, obtaining corpora with NL requirements and their temporal formalization containing millions of words is impractical.

V. Related work

In [40], a review of requirements formalization approaches is given. The majority of considered target formalisms have no relation to temporal logics. The work [41] is a survey of a rather new direction in machine translation: the use of semantic web technologies (SWT) [42]. According to this work, SWT are powerful in the task of disambiguation. The report [43] studies NL processing approaches related to requirements formalization. In particular, it considers generation of ontologies and UML diagrams from NL requirements and the challenges of the formalization process.

VI. Discussion and conclusions

This paper has reviewed existing and potentially possible means of formalizing NL requirements into discrete time temporal logics, such as LTL and CTL. First, it has considered direct solutions of this problem, which are often based on formal grammars. These solutions are summarized in Table I. A couple of them have been found promising. In particular, the approach [23] converts requirements into SystemVerilog Assertions (a language similar to LTL), captures the writing style used in the training corpus, and does not require much training data. Common shortcomings of reviewed translation approaches are restrictions on the grammatical structure of NL and handling its ambiguity.

Second, the problem of converting NL into a temporal logic (that is, to a formal language) has been found to be to some extent analogous to the one of translating NLs between each other. Unfortunately, machine translation requires large parallel corpora to be effective. This is especially crucial for neural machine translation, which needs millions of words in the training corpus to outperform traditional statistical approaches. The latter, although being less demanding, are also less precise in translation; this would lead to the need of manual editing of temporal requirements after their generation.

In the problem of requirements formalization into a temporal logic, the temporal logic part of the corpus must be created by experts, meaning that the task of collecting a

TABLE I
COMPARISON OF DIRECT NL REQUIREMENTS FORMALIZATION APPROACHES

Approach	Target temporal logic	NL restrictions	Approach to ambiguity resolution
[11]	LTL	NL requirements must follow the structure of predefined patterns [13], although vocabulary and specification style can be customized according to a domain.	Since the patterns [13] have explicit meanings in LTL, ambiguities are impossible.
[17]	ACTL (CTL with actions)	NL requirements must follow a fixed context-free grammar constructed to represent ACTL using English sentences. Concrete words used in a domain can be specified in a dictionary.	If an ambiguity is detected, the user is asked to provide the missing information.
[18]	ACTL (CTL without temporal operators starting with **E**)	More flexible than in [17], but the language still needs to be precise and concrete. Multi-sentence requirements are supported.	Multiple alternative translations may be generated. The user may be asked to resolve the meaning of pronouns and some other words.
[20]	LTL	NL requirements must follow a fixed context-free grammar constructed to represent tasks for a mobile robot.	The NL grammar is constructed in a way to avoid ambiguities.
[21]	CTL*	NL requirements must follow a fixed combinatorial categorial grammar. Requirements must be complete and grammatical.	The ambiguity problem is discussed (e.g., lexically ambiguous words are possible), but not essentially solved.
[22]	CTL	NL requirements must follow an attribute grammar (essentially an extension of a context-free grammar) that is created manually based on a corpus of requirements.	Some information can be lost during translation. New translation rules can be added manually based on the corpus.
[23]	SystemVerilog assertions	NL requirements must follow an attribute grammar that is learned automatically from a corpus of requirements and can capture the writing style.	The ambiguity problem is not mentioned, and in particular the authors do not report any translation ambiguity in experimental results.
[26]	LTL	NL requirements can be represented as arbitrary sentences, but only a predefined set of temporal patterns for the mobile robots domain can be recognized.	Since only a small number of temporal patterns may be recognized, requirements that are formulated not as intended or are grammatically ambiguous may be omitted.
[27]	LTL	More flexible than in [17], but the language still needs to be precise and concrete. Only present, future and passive tenses are supported.	The NL grammar is constructed in a way to avoid ambiguities.
[28]	LTL	The language needs to be precise and concrete. On the other hand, different domains and writing styles can be supported by customizing the translator.	Some information can be lost during translation. The translator can be customized manually (e.g., type rules can be extended) based on the corpus.

corpus becomes difficult already for thousands of words, not millions. On the other hand, requirements are written in a less ambiguous and more concise language compared to common NL. The vocabulary used in requirement documents is also smaller. Moreover, since requirements are often well-structured, artificial *data augmentation* with sentences derived from existing ones may be applied efficiently.

To sum up, direct approaches were found to be the most promising for the problem of requirements formalization into temporal logics. Among them, [23] and [28] stand out due to their flexibility: while the approach [23] learns NL representation based on a corpus, the work [28] proposes a rather general technique that can be customized for a particular domain. In contrast, while statistical approaches [29], [39] in theory can be adapted to solve this problem, they are likely to perform poorly due to the lack of large corpora of NL requirements paired with their temporal formalization.

A significant limitation of this study is the lack of an experimental evaluation of different approaches on the same parallel corpora. Unfortunately, such an evaluation cannot be performed for the majority of reviewed approaches since they depend on their domains of application and particular corpora of requirements. Also, generation of temporal requirements for object models [44], hybrid systems [45], and generation of first-order logic requirements [46] remained out of scope of this study.

A possible direction of future work is to focus on the approach [23]: examine the possibility of its generalization to support various target temporal logics and evaluate this approach on more diverse sets of requirements. In addition, both approaches [23] and [28] can be evaluated on requirements from the nuclear I&C domain.

ACKNOWLEDGMENTS

This work was financially supported by the SAUNA project (funded by the Finnish Nuclear Waste Management Fund VYR as a part of research program SAFIR2018), and by the Government of the Russian Federation (Grant 08-08).

REFERENCES

[1] E. M. Clarke, O. Grumberg, and D. Peled, *Model checking*. MIT press, 1999.

[2] C. Baier and J.-P. Katoen, *Principles of model checking*. MIT press, 2008.

[3] A. Pakonen, T. Tahvonen, M. Hartikainen, and M. Pihlanko, "Practical applications of model checking in the Finnish nuclear industry," in *10th International Topical Meeting on Nuclear Plant Instrumentation, Control and Human Machine Interface Technologies (NPIC & HMIT)*. American Nuclear Society, 2017, pp. 1342–1352.

[4] G. J. Holzmann, "The logic of bugs," *ACM SIGSOFT Software Engineering Notes*, vol. 27, no. 6, pp. 81–87, 2002.

[5] P. Koehn, F. J. Och, and D. Marcu, "Statistical phrase-based translation," in *2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology, Volume 1*. Association for Computational Linguistics, 2003, pp. 48–54.

[6] J. Bengtsson, K. Larsen, F. Larsson, P. Pettersson, and W. Yi, "UPPAAL – a tool suite for automatic verification of real-time systems," in *International Hybrid Systems Workshop*. Springer, 1995, pp. 232–243.

[7] A. Cimatti, E. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella, "NuSMV 2: An opensource tool for symbolic model checking," in *International Conference on Computer Aided Verification*. Springer, 2002, pp. 359–364.

[8] G. J. Holzmann, "The model checker SPIN," *IEEE Transactions on Software Engineering*, vol. 23, no. 5, pp. 279–295, 1997.

[9] C. Pang, A. Pakonen, I. Buzhinsky, and V. Vyatkin, "A study on user-friendly formal specification languages for requirements formalization," in *2016 IEEE 14th International Conference on Industrial Informatics (INDIN)*. IEEE, 2016, pp. 676–682.

[10] J. E. Hopcroft, R. Motwani, and J. D. Ullman, "Introduction to automata theory, languages, and computation," *Acm Sigact News*, vol. 32, no. 1, pp. 60–65, 2001.

[11] S. Konrad and B. H. Cheng, "Automated analysis of natural language properties for UML models," in *International Conference on Model Driven Engineering Languages and Systems*. Springer, 2005, pp. 48–57.

[12] M. B. Dwyer, G. S. Avrunin, and J. C. Corbett, "Property specification patterns for finite-state verification," in *2nd Workshop on Formal methods in Software Practice*. ACM, 1998, pp. 7–15.

[13] ——, "Patterns in property specifications for finite-state verification," in *21st International Conference on Software Engineering (ICSE)*. ACM, 1999, pp. 411–420.

[14] S. Flake, W. Müller, and J. Ruf, "Structured English for model checking specification," in *MBMV*, 2000, pp. 99–108.

[15] S. Konrad and B. H. Cheng, "Real-time specification patterns," in *Proceedings of the 27th international conference on Software engineering*. ACM, 2005, pp. 372–381.

[16] J. Dick, E. Hull, and K. Jackson, *Requirements engineering*. Springer, 2017.

[17] A. Fantechi, S. Gnesi, G. Ristori, M. Carenini, M. Vanocchi, and P. Moreschini, "Assisting requirement formalization by means of natural language translation," *Formal Methods in System Design*, vol. 4, no. 3, pp. 243–263, 1994.

[18] R. Nelken and N. Francez, "Automatic translation of natural language system specifications into temporal logic," in *International Conference on Computer Aided Verification*. Springer, 1996, pp. 360–371.

[19] H. Kamp, "A theory of truth and semantic representation," in *Formal Methods in the Study of Language*, J. Groenendijk and M. Stokhof, Eds. Mathematical Center, Amsterdam, 1981, pp. 277–322.

[20] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Translating structured English to robot controllers," *Advanced Robotics*, vol. 22, no. 12, pp. 1343–1359, 2008.

[21] J. Dzifcak, M. Scheutz, C. Baral, and P. Schermerhorn, "What to do and how to do it: Translating natural language directives into temporal and dynamic logic representation for goal management and action execution," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2009, pp. 4163–4168.

[22] C. B. Harris and I. G. Harris, "Generating formal hardware verification properties from natural language documentation," in *2015 IEEE International Conference on Semantic Computing (ICSC)*. IEEE, 2015, pp. 49–56.

[23] ——, "GLAsT: Learning formal grammars to translate natural language specifications into hardware assertions," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2016, pp. 966–971.

[24] D. Bustan and J. Havlicek, "Some complexity results for SystemVerilog assertions," in *International Conference on Computer Aided Verification (CAV)*. Springer, 2006, pp. 205–218.

[25] A. D'Ulizia, F. Ferri, and P. Grifoni, "A survey of grammatical inference methods for natural language learning," *Artificial Intelligence Review*, vol. 36, no. 1, pp. 1–27, 2011.

[26] V. Raman, C. Lignos, C. Finucane, K. C. Lee, M. P. Marcus, and H. Kress-Gazit, "Sorry Dave, I'm afraid I can't do that: Explaining unachievable robot tasks using natural language," in *Robotics: Science and Systems*, vol. 2, no. 1, 2013.

[27] R. Yan, C.-H. Cheng, and Y. Chai, "Formal consistency checking over specifications in natural languages," in *2015 Design, Automation & Test in Europe Conference & Exhibition*. EDA Consortium, 2015, pp. 1677–1682.

[28] S. Ghosh, D. Elenius, W. Li, P. Lincoln, N. Shankar, and W. Steiner, "ARSENAL: automatic requirements specification extraction from natural language," in *NASA Formal Methods Symposium*. Springer, 2016, pp. 41–46.

[29] P. Koehn, *Statistical machine translation*. Cambridge University Press, 2009.

[30] E. Alpaydin, *Introduction to machine learning*. MIT press, 2009.

[31] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "BLEU: a method for automatic evaluation of machine translation," in *40th Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 2002, pp. 311–318.

[32] A. P. Nikora and G. Balcom, "Automated identification of LTL patterns in natural language requirements," in *2009 20th International Symposium on Software Reliability Engineering*. IEEE, 2009, pp. 185–194.

[33] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information Processing & Management*, vol. 24, no. 5, pp. 513–523, 1988.

[34] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.

[35] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, p. 533, 1986.

[36] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in Neural Information Processing Systems*, 2014, pp. 3104–3112.

[37] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[38] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *Journal of Machine Learning Research*, vol. 3, no. Feb, pp. 1137–1155, 2003.

[39] P. Koehn, "Neural machine translation," *arXiv preprint arXiv:1709.07809*, 2017.

[40] S. Vadera and F. Meziane, "Tools for producing formal specifications: a view of current architectures and future directions," *Annals of Software Engineering*, vol. 3, no. 1, pp. 273–290, 1997.

[41] D. Moussallem, M. Wauer, and A.-C. N. Ngomo, "Machine translation using semantic web technologies: A survey," *Journal of Web Semantics*, vol. 51, pp. 1–19, 2018.

[42] P. Hitzler, M. Krotzsch, and S. Rudolph, *Foundations of semantic web technologies*. Chapman and Hall/CRC, 2009.

[43] T. Bures, P. Hnetynka, P. Kroha, and V. Simko, "Requirement specifications using natural languages," Technical Report D3S-TR-2012-05. Czechoslovakia: Charles University in Prague, Tech. Rep., 2012.

[44] A. Cimatti, M. Roveri, A. Susi, and S. Tonetta, "Formalizing requirements with object models and temporal constraints," *Software & Systems Modeling*, vol. 10, no. 2, pp. 147–160, 2011.

[45] ——, "Validation of requirements for hybrid systems: A formal approach," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 21, no. 4, p. 22, 2012.

[46] A. Ranta, "Translating between language and logic: what is easy and what is difficult," in *International Conference on Automated Deduction*. Springer, 2011, pp. 5–25.