Towards Explainable Formal Methods: from LTL to Natural Language with Neural Machine Translation

Himaja Cherukuri[1], Alessio Ferrari[2][0000−0002−0636−5663], and Paola Spoletini[1][0000−0001−7922−4936]

[1] Kennesaw State University, Atlanta (GA), USA
chhimajasri@gmail.com, pspoleti@kennesaw.edu
[2] CNR-ISTI, Pisa, Italy, alessio.ferrari@isti.cnr.it

Abstract. [Context and motivation] Requirements formalisation facilitates reasoning about inconsistencies, detection of ambiguities and identification critical issues in system models. Temporal logic formulae are the natural choice when it comes to formalise requirements associated to desired system behaviours. **[Question/problem]** Understanding and mastering temporal logic requires a formal background. Means are therefore needed to make temporal logic formulae interpretable by engineers, domain experts and other stakeholders involved in the development process. **[Principal ideas/results]** In this paper, we propose to use a neural machine translation tool, named OpenNMT, to translate Linear Temporal Logic (LTL) formulae into corresponding natural language descriptions. Our results show that our translation system achieves an average BLEU (BiLingual Evaluation Understudy) score of 93.53%, which corresponds to high-quality translations. **[Contribution]** Our neural model can be applied to assess if requirements have been correctly formalised. This can be useful to requirements analysts, who may have limited confidence with LTL, and to other stakeholders involved in the requirements verification process. Overall, our research preview contributes to bridging the gap between formal methods and requirements engineering, and opens to further further research in explainable formal methods.

Keywords: Requirements engineering · Formal Methods · Machine Translation · Neural Networks · Temporal Logic · LTL.

1 Introduction

Temporal logic enables the expression of time-related system requirements and has widely been used in requirements and software engineering research [20, 5]. Linear temporal logic (LTL) is a well-known type of temporal logic that treats time as a linear sequence of states. LTL has been used in requirements engineering (RE) for several tasks, including the formalization of goals in goal-oriented requirements engineering (GORE) [15], the expression of desired properties for run-time verification [3], and model checking [6]. The correct specification and interpretation of LTL formulae requires a strong mathematical background and can hardly be done by domain experts [5,

7]. Therefore, researchers have dedicated efforts to translate natural language (NL) requirements into temporal logic formulae [17, 10, 5, 11] to support domain experts in the formalization of requirements. However, these approaches still require domain experts to have an understanding of the produced formulae, so to make sure that the translation is correctly preserving the meaning of the original requirement. To support them in this task, we propose to provide a way to translate LTL formulae into their NL explanation. To address this goal, we plan to exploit the potential of neural machine translation platforms, and in particular the open-source framework OpenNMT (`https://opennmt.net`). Indeed, though the goal of translating LTL into corresponding explanations can in principle be addressed by means of a rule-based or heuristic approach, a neural machine translation strategy is more flexible, as it can facilitate language simplification and transformations—i.e., summaries and paraphrases [13, 19, 2], without requiring the maintenance of a complex rule-based system. In addition, it can better support the readability of the expressions [2], while ensuring the correctness of the translation. As LTL formulae can often be better understood when associated with visual representations [1], we also plan to augment the translation with a graphical representation that could help clarifying possible ambiguities introduced by the NL translation. At the current stage, we have performed a feasibility study, in which we trained an LTSTM encoder-decoder architecture, implemented in the OpenNMT framework, with a set of manually defined examples. In the next steps, we will consolidate the approach, we will develop the visual part of our idea, and we will validate the resulting prototype with potential users.

2 Towards Explainable LTL Requirements

The overall goal of our research is to facilitate the correct understanding of requirements expressed in LTL by subjects who have a limited expertise in formal logic. To this end, we plan to implement a system that translates LTL formulae into corresponding NL explanations, augmented by visual diagrams with annotations. We will also empirically evaluate the approach, first by ensuring that the automatic translation is actually correct, and then by evaluating to what extent the translation facilitates the understanding of LTL requirements. More specifically, our research questions (RQs) are the following:

- **RQ1:** *To what extent can neural machine translation be used to translate LTL formulae into NL explanations?*
- **RQ2:** *How can NL explanations of LTL formulae be augmented with visual representations?*
- **RQ3:** *Does the automatic explanation of LTL formulae help users in understanding them?*

To answer RQ1, we first perform a feasibility study, reported in this paper (cf. Sect. 2.1), and then we consolidate the approach by (a) ensuring that the approach does not introduce errors in the translation, and (b) ensuring that the readability of the formulae is acceptable, according to standard metrics and through human assessment. To answer RQ2, we plan to devise solutions by combining visual representation of formulae and annotation of traces. Finally, RQ3 will be addressed through an empirical evaluation with students.

2.1 RQ1: From LTL to NL with Neural Machine Translation

Dataset Definition To assess the feasibility of using neural machine translation for providing explanation of LTL formulae, the 3rd author defined 54 unique formulae including Boolean operators ($!, |, \&, =>$) and temporal operators (X—next, G—always, F—eventually, U—until), with associated NL translations. The 2nd author independently checked the correctness of the translation. For simplicity, the dataset considers only formulae with no nested temporal operators and the expressions are edited according to the typical LTL patterns as defined by Dwyer et al. [8], so as to provide representative LTL requirements that could occur in real-world projects. This initial dataset is composed of domain-agnostic requirements, in which variables were expressed as alphabetic letters, e.g. $G(a => b)$, translated as *In every moment of the system execution, if a holds, b should hold (a does not need to hold for the formula to be true)*. In providing the translations, the 3rd author made an effort to be consistent across formulae, using always the same terminology, and the same structure. However, no translation rule was established beforehand. The repetitiveness of terminology and structure aims to facilitate learning of the neural model, while the absence of specific rules decided beforehand enables flexibility. As the set of examples would be too limited for successfully training a neural network, the dataset was clerically augmented, by repeating the same formulae—and associated translations—with combinations of 26 different alphabetic letters. The resulting dataset is composed of 12,192 LTL formulae and associated translations. At this stage, our goal is not to translate unseen syntactic pattern, but to check whether the unwritten rules adopted for translation can be successfully learned. Therefore, we feed the network with similar examples that differ solely for the variable names. The idea is to enable the network to distinguish between operators and variables, "learn" the LTL syntax for these simple cases, and translate accordingly.

Training and Evaluation To experiment with our dataset, we selected the OpenNMT framework for machine translation [14]. This is a widely used platform, supporting different translation tasks, including summarization, image to text, speech-to-text, sequence-to-sequence (our case), and offering two implementations, OpenNMT-py, based on PyTorch, and OpenNMT-tf, based on TensorFlow. In our case, we selected OpenNMT-py, as it is claimed by the developers to be more user-friendly, and thus we consider it more appropriate for the exploratory nature of our study. The architecture adopted for the task is a 2-layer Long short-term memory (LSTM) neural network with 500 hidden units on both the encoder and decoder. This is a recurrent neural network (RNN) often used for sequence-to-sequence learning. We use the default settings of OpenNMT at this stage, given the exploratory nature of the study.

We randomly split our dataset into training (8,048 items, 66% of the total), validation (1,836, 15%) and test set (2,308, 19%). The validation set is used to evaluate the convergence of the training. The model that achieves the lowest perplexity value on this dataset is considered the best and selected for evaluation in the test set. To avoid oversimplifying the problem, the test set includes only cases with more than one variable, i.e., more than one alphabetic letter. The whole training activity lasted 7.8 hours on a common laptop.

We evaluate the results on the test set by means of different metrics to check the quality of the translation. Evaluation is carried out by means of the Tilde MT online tool (https://www.letsmt.eu/Bleu.aspx). The readability of the resulting formulae is assessed with the BLEU score [18]. The visual representation provided by Tilde MT is used to identify translations with BLEU score lower than 100%—suggesting incorrect translations—and manually assess them. Indeed, here we want to ensure that the translation is actually 100% correct, and while a high BLEU score between expected and translated sentence could indicate high similarity, the actual difference (e.g., in terms of variable names, or in case a negation is missing) could be crucial for the correctness of the translation.

The BLEU score is 93.53%, indicating high-quality translations, thus suggesting that the translation of LTL formulae with neural machine translation is feasible. It is worth noting that issues are known with the usage of automatic scores in machine translation applied to software engineering problems [12], and further studies with human subjects need to be performed to actually assess the quality of the translation.

Looking at single cases with lower BLEU score, we see that while the syntax is somehow correct, there are some difficulties with the U operator. For example, the formula $(c \,\&\, q\,)\; U\; o$ is translated as *There has to be a moment (the current one or in the future) in which u holds, and, if it is not the current one, from the current moment to that moment both c and q have to hold*, BLEU = 94%. The translator introduces the a spurious u variable, possibly confused by the letter U of the operator. Similar situations however occur also with other letters. Low BLEU scores are obtained also for complex expressions such as $(\; c\; U\; q\;) \,\&\, (\; o\; U\; q\;)$, in which only the initial part of the formula is translated, while the second part is entirely missing: *There has to be a moment (the current one or in the future) in which q holds, and, if it is not the current one, from the current moment to that moment c has to hold*, BLEU = 35.9%. The first issue could be addressed by using specific keywords or characters for the operators, or experimenting with longer translation units (i.e., words). The second problem could be solved by segmenting the formula beforehand with rule-based approaches before feeding it to the translator.

Consolidation The preliminary evaluation carried out suggests that the project idea is feasible with currently available technologies. Further work is required, however, to provide empirically sound evidence to answer RQ1. In particular, besides replicating the current experiments with different neural network architectures, the next steps of our research will address the issue of correctness, by studying the possible problems leading to inaccuracy, and providing solutions towards the goal of 100% correctness [4]. Furthermore, we will extend the evaluation to nested operators, so that full coverage of LTL formulae is possible. Concerning readability of the translations, we plan to work in two directions. The first one consists in assessing the readability of the translations in the context of the experiments with human subjects carried out in relation to RQ3 (cf. Sect. 2.2). The second direction aims to enhance the approach with automatic text simplification techniques [2], which can be particularly useful in case of lengthy and hard-to-process translations.

2.2 RQ2, RQ3: Visual Representations and Empirical Evaluation

The research activities related to RQ2 and RQ3 will be carried out in parallel with the consolidation of the results of RQ1.

In relation to **RQ2**, we will first investigate possible solutions to augment LTL explanations with visual information. This investigation will consider both the graphical representation of the formulae, in line with e.g., Ahmed *et al.* [1], the representation of the associated traces as done by the LTL Visualiser tool (`https://quickstrom.github.io/ltl-visualizer/`), and the annotation of traces with NL text generated from the formulae. To select the appropriate means for graphical representation of formulae, we will follow a design science approach [21]. Stemming from the literature, we will design an innovative prototypical solution, and we will perform iterations to refine and validate it. Differently from the deep learning-based translation of LTL formulae, the graphical representation is expected to leverage a rule-based algorithm. Therefore, its correctness is to be ensured by construction—provided that systematic tests against the requirements are carried out.

To answer **RQ3**, we will conduct a controlled experiment to measure if the generated explanations improve the understandability of LTL formulae. The experiment will be run with senior undergraduate students and graduate students attending an RE course covering temporal logic. Participants of control group and experimental one will be given a set of LTL formulae. For each formula, they will be also given a set of traces, and their task will be to select all the traces that satisfy the given formula—this exercise is regarded as a way to assess their correct understanding of the formula. In addition to the formulae, the experimental group will also be given as input the textual and graphical explanation for each of the formulae, as generated by our approach. Checking the performance of the two groups will allow us to measure the quality of the support provided by our solution in this activity. After the activity, participants will be asked to fill out a questionnaire to gather their perceptions about the task and, for the experimental group, the support obtained by the explanation. To support evaluation of readability, we plan to also repeat the experiment with eye-tracking devices.

3 Conclusion

This paper presents a research preview on providing means to make requirements expressed through LTL understandable to subjects with limited expertise in formal logic. The proposed approach has the potential to be a useful tool to support students and practitioners in learning LTL, but can also have applications in practice. For example, it can facilitate mutual understanding in those industry-academia collaborations in which practitioners provide the informal system specification, and formal methods experts provide formal designs, as common, e.g. in the railway domain [9]. Furthermore, the approach can be used to support verification via model checking of incomplete systems, which is needed when a software is developed incrementally or through decomposition. Existing solutions to this problem (e.g., Menghi *et al.* [16]) rely on the generation of LTL constraints to be satisfied by novel components to be developed. In these contexts, NL explanations can be particularly useful to requirements analysts and developers in the design of the novel components. Our research is also among the first

attempts to integrate deep learning with the world of formal methods, possibly fostering cross-fertilisation between the fields.

References

1. Ahmed, Z., Benque, D., Berezin, S., Dahl, A.C.E., Fisher, J., Hall, B.A., Ishtiaq, S., Nanavati, J., Piterman, N., Riechert, M., et al.: Bringing LTL model checking to biologists. In: VMCAI. pp. 1–13. Springer (2017)
2. Al-Thanyyan, S.S., Azmi, A.M.: Automated text simplification: A survey. ACM Computing Surveys (CSUR) **54**(2), 1–36 (2021)
3. Bauer, A., Leucker, M., Schallhart, C.: Runtime verification for LTL and TLTL. ACM Transactions on Software Engineering and Methodology (TOSEM) **20**(4), 1–64 (2011)
4. Berry, D.M.: Empirical evaluation of tools for hairy requirements engineering tasks. Empir. Softw. Eng. **26**(5), 111 (2021)
5. Brunello, A., Montanari, A., Reynolds, M.: Synthesis of LTL formulas from natural language texts: State of the art and research directions. In: 26th International Symposium on Temporal Representation and Reasoning (TIME 2019) (2019)
6. Clarke, E., Grumberg, O., Kroening, D., Peled, D., Veith, H.: Model Checking. Cyber Physical Systems Series, MIT Press (2018)
7. Czepa, C., Zdun, U.: On the understandability of temporal properties formalized in linear temporal logic, property specification patterns and event processing language. IEEE Transactions on Software Engineering **46**(1), 100–112 (2018)
8. Dwyer, M.B., Avrunin, G.S., Corbett, J.C.: Patterns in property specifications for finite-state verification. In: ICSE'99. p. 411–420 (1999)
9. Ferrari, A., ter Beek, M.H.: Formal methods in railways: a systematic mapping study (2021), https://arxiv.org/abs/2107.05413
10. Ghosh, S., Elenius, D., Li, W., Lincoln, P., Shankar, N., Steiner, W.: Arsenal: automatic requirements specification extraction from natural language. In: NASA Formal Methods Symposium. pp. 41–46. Springer (2016)
11. Giannakopoulou, D., Pressburger, T., Mavridou, A., Schumann, J.: Automated formalization of structured natural language requirements. IST **137**, 106590 (2021)
12. Gros, D., Sezhiyan, H., Devanbu, P., Yu, Z.: Code to comment "translation": Data, metrics, baselining & evaluation. In: ASE 2020. pp. 746–757. IEEE (2020)
13. Gupta, A., Agarwal, A., Singh, P., Rai, P.: A deep generative framework for paraphrase generation. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 32 (2018)
14. Klein, G., Kim, Y., Deng, Y., Senellart, J., Rush, A.: OpenNMT: Open-source toolkit for neural machine translation. In: ACL 2017. pp. 67–72 (2017)
15. Letier, E., Kramer, J., Magee, J., Uchitel, S.: Deriving event-based transition systems from goal-oriented requirements models. Automated Software Engineering **15**(2), 175–206 (2008)
16. Menghi, C., Spoletini, P., Chechik, M., Ghezzi, C.: Supporting Verification-Driven Incremental Distributed Design of Components. In: FASE 2018. pp. 169–188. Springer (2018)
17. Nikora, A.P., Balcom, G.: Automated identification of LTL patterns in natural language requirements. In: ISSRE'09. pp. 185–194. IEEE (2009)
18. Papineni, K., Roukos, S., Ward, T., Zhu, W.J.: Bleu: a method for automatic evaluation of machine translation. In: ACL'02. pp. 311–318 (2002)
19. Siddharthan, A.: A survey of research on text simplification. ITL-International Journal of Applied Linguistics **165**(2), 259–298 (2014)
20. Van Lamsweerde, A., Letier, E.: Handling obstacles in goal-oriented requirements engineering. IEEE Transactions on software engineering **26**(10), 978–1005 (2000)
21. Wieringa, R.J.: Design science methodology for information systems and software engineering. Springer (2014)