| **Toward Auto-Modeling of Formal Verification for**  **NextG Protocols: A Multimodal Cross- and**  **Self-Attention Large Language**  **Model Approach** | This paper introduces Auto-modeling of Formal Verification with Real-world Prompting for  5G and NextG protocols (AVRE), a novel system designed for the formal verification of Next Generation  (NextG) communication protocols, addressing the increasing complexity and scalability challenges in  network protocol design and verification. Utilizing Large Language Models (LLMs), AVRE transforms  protocol descriptions into dependency graphs and formal models, efficiently resolving ambiguities and  capturing design intent. The system integrates a transformer model with LLMs to autonomously establish  quantifiable dependency relationships through cross- and self-attention mechanisms. Enhanced by iterative  feedback from the HyFuzz experimental platform, AVRE significantly advances the accuracy and relevance  of formal verification in complex communication protocols, offering a groundbreaking approach to  validating sophisticated communication systems. We compare CAL’s performance with state-of-the-art  LLM-based models and traditional time sequence models, demonstrating its superiority in accuracy and  robustness, achieving an accuracy of 95.94% and an AUC of 0.98. This NLP-based approach enables, for the  first time, the creation of exploits directly from design documents, making remarkable progress in scalable  system verification and validation. | CONVERTING INFORMAL NATURAL LANGUAGE SYSTEM  DESIGNS AND PROTOCOLS TO FORMAL DESCRIPTION  Approaches to transforming natural language descriptions  into formal models have seen considerable advancements,  evolving through the introduction of diverse methodologies  over the years. A decade ago, Drechsler et al. [13] proposed  a paradigm that incorporated the Formal Specification Level  (FSL), adeptly bridging the gap between informal textbook  specifications and formal Electronic System Level (ESL)  interpretations. Subsequently, Banarescu et al. [14] proposed  a hybrid methodology that converted linguistic expressions  into formal paradigms by merging symbolic and statistical  techniques. With the advent of deep learning, Dong and  Lapata [15] employed neural networks to convert natural  language instructions into executable codes. This work was  further enhanced by the contributions of Reddy et al. [16] in  2019, who focused on semantic parsing, utilizing denotations  to transform complex linguistic structures into formalized  notations. Meanwhile, the application of regular expressions  has been identified as a viable means to extract formal  specifications from natural language narratives, providing advanced components in deep learning frameworks [17].  Despite these developments, models based on these previous  methodologies have achieved an accuracy threshold of  approximately 90%, which is inadequate for ensuring com-  plete recall. This poses a challenge in the precise conversion  of natural language protocols into formal formulations.  2) LLMS BASED FORMAL VERIFICATION  LLMs have demonstrated impressive reasoning and assertion  capabilities for formal verification [9], [18], [19]. Research  in [8] and [9] has explored using LLMs to generate temporal  logic specifications and assertions from unstructured natural  languages. Meanwhile, studies in [20] and [21] focus on  leveraging LLMs to enhance BMC for identifying software  vulnerabilities and deriving counterexamples. In [10], the  authors trained GPT-4 to generate correct SystemVerilog  Assertions (SVA) through iterative prompt refinement with  rules. However, it remains unclear how these models derive  answers and whether they rely on simple heuristics rather than  a generated chain-of-thought [18]. The current state of the  art prioritizes producing formal specifications and properties  quickly, albeit with slight inaccuracies, over generating  perfect specifications or correctness statements [19]. The  non-transparency related to LLM heuristics leads to a  large number of irrelevant dependencies, resulting in low  precision in dependency classification. To address this, our  experimental platform connects to guide and refine the  dependency graph range.  3) PROMPTING LIMITATIONS IN LLM ENABLED FORMAL  VERIFICATION  Furthermore, the majority of existing work relies on prompt  engineering. LLM-integrated applications blur the line  between data and instructions [22]. LLMs can produce non-  deterministic outputs, potentially yielding different results for  the same prompt. This variability poses a potential threat  to the validity of scientific conclusions unless researchers  adapt their methods to account for it in their empirical  analyses [23]. The adoption of prompting methods introduces  challenges in iterative formal verification without human  involvement. The randomness in LLMs is influenced by the  sampling methods used during text generation, such as top-k  sampling or nucleus sampling [24], limiting its application in  classifiers or deterministic types of applications. To address  the non-determinism and iterative formal verification,  distinct from current formal verification methods that utilize  prompt engineering, we designed an open-access LLM,  integrated with a transformer model, to achieve supervised  dependency.’’  4) DIGITAL ENGINEERING AIDED FORMAL GUIDED SYSTEM  VALIDATION  In the field of integrated Design Validation, there has been  recent research progress in combining formal verification  with simulation, resulting in a practical validation engine  27860  with reasonable run-time [25]. Experimental work in the  context of 5G has gained significant attention over the  past few years, shifting from the simulation-driven research  used in previous mobile network generations to system  implementation prototyping [26]. In our previous work,  we explored a Formal-guided Fuzzing testing approach  [27], [28] to bridge design verification and system validation.  This approach complements the scalability limitations of  formal verification and addresses the impacts of detected  vulnerabilities. We have introduced a fuzzing digital twin [29]  to provide an open and automated platform for systematically  that enables an autonomous detection of vulnerabilities  and unintended emergent behaviors in 5G infrastructures.  However, a constraint of this initial approach is heavy  reliance on expert insights to identify and articulate formal  relationships. |
| --- | --- | --- |

[8] R. Kande, H. Pearce, B. Tan, B. Dolan-Gavitt, S. Thakur, R. Karri, and J. Rajendran, ‘‘LLM-assisted generation of hardware assertions,’’ 2023, arXiv:2306.14027.

[9] M. Cosler, C. Hahn, D. Mendoza, F. Schmitt, and C. Trippel, ‘‘Nl2spec: Interactively translating unstructured natural language to temporal logics with large language models,’’ 2023, arXiv:2303.04864.

[10] M. Orenes-Vera, M. Martonosi, and D. Wentzlaff, ‘‘Using LLMs to facilitate formal verification of RTL,’’ 2023, arXiv:2309.09437

[13] R. Drechsler, M. Soeken, and R. Wille, ‘‘Formal specification level: Towards verification-driven design based on natural language processing,’’ in Proc. Forum Specification Design Lang., Sep. 2012, pp. 53–58.

[14] L. Banarescu, C. Bonial, S. Cai, M. Georgescu, K. Griffitt, U. Hermjakob, K. Knight, P. Koehn, M. Palmer, and N. Schneider, ‘‘Abstract meaning representation for sembanking,’’ in Proc. 7th Linguistic Annotation Workshop Interoperability Discourse, 2013, pp. 178–186.

[15] L. Dong and M. Lapata, ‘‘Language to logical form with neural attention,’’ 2016, arXiv:1601.01280.

[16] S. Reddy, D. Chen, and C. D. Manning, ‘‘CoQA: A conversational question answering challenge,’’ Trans. Assoc. Comput. Linguistics, vol. 7, pp. 249–266, Nov. 2019.

[17] C. Hahn, F. Schmitt, J. J. Tillman, N. Metzger, J. Siber, and B. Finkbeiner, ‘‘Formal specifications from natural language,’’ 2022, arXiv:2206.01962.

[18] A. Saparov and H. He, ‘‘Language models are greedy reasoners: A systematic formal analysis of chain-of-thought,’’ 2022, arXiv:2210.01240.

[19] P. Srikumar, ‘‘Fast and wrong: The case for formally specifying hardware with LLMS,’’ in Proc. Int. Conf. Architectural Support Program. Lang. Operating Syst. (ASPLOS). New York, NY, USA: ACM Press, 2023.

[20] N. Tihanyi, T. Bisztray, R. Jain, M. A. Ferrag, L. C. Cordeiro, and V. Mavroeidis, ‘‘The FormAI dataset: Generative AI in so ware security through the lens of formal verification,’’ in Proc. PROMISE, 2023, p. 33.

[21] Y. Charalambous, N. Tihanyi, R. Jain, Y. Sun, M. Amine Ferrag, and L. C. Cordeiro, ‘‘A new era in software security: Towards self-healing software via large language models and formal verification,’’ 2023, arXiv:2305.14752.

[22] K. Greshake, S. Abdelnabi, S. Mishra, C. Endres, T. Holz, and M. Fritz, ‘‘Not what you’ve signed up for: Compromising real-world LLMintegrated applications with indirect prompt injection,’’ in Proc. 16th ACM Workshop Artif. Intell. Secur., Nov. 2023, pp. 79–90.

[23] S. Ouyang, J. M. Zhang, M. Harman, and M. Wang, ‘‘LLM is like a box of chocolates: The non-determinism of ChatGPT in code generation,’’ 2023, arXiv:2308.02828.

[24] K. Krishna, Y. Chang, J. Wieting, and M. Iyyer, ‘‘RankGen: Improving text generation with large ranking models,’’ 2022, arXiv:2205.09726.

[25] L. Li and M. Thornton, Digital System Verification: A Combined Formal Methods and Simulation Framework. Berlin, Germany: Springer Nature, 2022.

[26] Y. Wang, A. Gorski, and A. P. da Silva, ‘‘Development of a data-driven mobile 5G testbed: Platform for experimental research,’’ in Proc. IEEE Int. Medit. Conf. Commun. Netw. (MeditCom), Sep. 2021, pp. 324–329.

[27] J. Yang, S. Arya, and Y. Wang, ‘‘Formal-guided fuzz testing: Targeting security assurance from specification to implementation for 5G and beyond,’’ 2023, arXiv:2307.11247.

[28] J. Yang and Y. Wang, ‘‘Formal and fuzzing amplification: Targeting vulnerability detection in 5G and beyond,’’ 2023, arXiv:2307.05758.

[29] D. Dauphinais, M. Zylka, H. Spahic, F. Shaik, J. Yang, I. Cruz, J. Gibson, and Y. Wang, ‘‘Automated vulnerability testing and detection digital twin framework for 5G systems,’’ in Proc. IEEE 9th Int. Conf. Netw. Softwarization (NetSoft), Jun. 2023, pp. 308–310.