7. Conclusion

This article has examined the problem of determining the optimal block size in short packets for 5G and 6G URLLC systems. By focusing on conceptual analysis and finite blocklength principles, the study highlights the fundamental trade-offs between latency, reliability, and block size in short-packet communication.

The results indicate that optimal block size is highly dependent on channel conditions and application requirements, and that adaptive block size selection is a key enabler of reliable and low-latency communication. These insights provide valuable guidance for the design and optimization of future URLLC-capable wireless networks.

References

1. Polyanskiy, Y., Poor, H. V., & Verdú, S. Finite Blocklength Information Theory
2. Durisi, G., Koch, T., & Popovski, P. Short-Packet Communications
3. Popovski, P. Ultra-Reliable Low-Latency Communication
4. 3GPP. 5G NR Specifications for URLLC

DOI 10.70286/ISU-24.12.2025.005

НЕЙРОСИМВОЛЬНІ ПІДХОДИ В АНАЛІЗІ ПРОГРАМНОГО КОДУ: ОГЛЯД СУЧАСНОГО СТАНУ ТА ПЕРСПЕКТИВИ РОЗВИТКУ

Вохранов Ілля Анатолійович
аспірант
Кафедра системного проектування
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського», Україна

Сучасні програмні системи характеризуються зростаючою складністю, що супроводжується підвищеним ризиком виникнення дефектів та вразливостей. За даними Національної бази даних вразливостей (NVD), кількість зареєстрованих вразливостей зросла з 7928 у 2014 році до понад 40000 у 2024 році, що становить п'ятикратне збільшення протягом десяти років [1]. Це актуалізує потребу в ефективних методах забезпечення якості та безпеки програмного забезпечення.

Традиційні методи статичного аналізу, такі як абстрактна інтерпретація та символьне виконання, забезпечують формальні гарантії коректності, проте стикаються з обмеженнями масштабованості та високим рівнем хибнопозитивних спрацювань [2]. Паралельно спостерігається інтенсивний розвиток методів машинного навчання для аналізу коду, які демонструють здатність виявляти складні патерни, але часто не надають формальних гарантій та характеризуються непрозорістю прийняття рішень.

Нейросимвольний штучний інтелект (Neuro-Symbolic AI) виникає як перспективна парадигма, що прагне поєднати переваги обох підходів: здатність

нейронних мереж до навчання та розпізнавання складних патернів із формальною строгістю символічного розмірковування [3]. Систематичний огляд нейросимвольного ШІ у 2024 році виявив 167 релевантних публікацій із відкритими кодовими базами, що свідчить про зростаючий інтерес до цієї області [4].

Символічний ШІ (Symbolic AI) базується на маніпуляції явними символічними представленнями знань через формальні правила логічного виведення. Його перевагами є інтерпретованість, здатність до точного розмірковування та можливість інтеграції експертних знань. Однак символічні системи вимагають значних зусиль для ручного кодування знань та погано справляються з невизначеністю та шумом у даних [5]. Субсимвольний ШІ (Sub-Symbolic AI), зокрема глибоке навчання, автоматично навчається представленням із даних та демонструє видатну здатність до узагальнення. Водночас нейронні моделі часто функціонують як «чорні скриньки», що ускладнює їх верифікацію та інтерпретацію [6].

Нейросимвольна інтеграція може бути реалізована через різні парадигми. Marra et al. [7] виділяють три основні категорії: логічно-інформовані підходи до навчання представлень, підходи з логічними обмеженнями та підходи до навчання правил. У контексті аналізу програмного коду особливого значення набувають гібридні архітектури, що поєднують нейронні моделі з традиційними інструментами статичного аналізу (табл. 1).

Таблиця 1. Таксономія нейросимвольних підходів до аналізу коду

Категорія	Характеристика	Приклади
Логічно-інформовані представлення	Символічні знання керують процесом навчання нейронних представлень коду	GraphCodeBERT, CodeT5
Навчання з логічними обмеженнями	Логічні правила формулюються як обмеження під час навчання моделі	Scallop, DeepProbLog
Нейронне навчання правил	Нейронні мережі автоматично виводять символічні правила з даних	Нейронний синтез програм
Гібридні архітектури	Послідовне поєднання LLM та статичних аналізаторів	IRIS, LLift

Розвиток попередньо навчених моделей для програмного коду став важливою віхою в цій галузі. CodeBERT [8] є бімодальною моделлю для програмних та природних мов. GraphCodeBERT [9] розширює цей підхід, враховуючи семантичну структуру коду через потік даних. CodeT5 [10] та UniXcoder [11] демонструють подальшу еволюцію, забезпечуючи покращення на задачах пошуку коду, виявлення клонів, перекладу та рефакторингу коду, створюючи основу для нейросимвольної інтеграції.

Великі мовні моделі (Large Language Models, LLM) відкрили нові можливості для взаємодії з програмним кодом. Систематичний огляд застосування LLM у кібербезпеці [12] виявив значний потенціал цих моделей для виявлення вразливостей. Дослідження демонструють, що LLM можуть ефективно використовуватися для підвищення якості перевірки коду, особливо у виявленні проблем безпеки. Водночас LLM мають суттєві обмеження: вони не здатні виконувати складне розмірковування над кодом на рівні репозиторію та можуть генерувати некоректні відповіді («галюцинації»). Це мотивує розробку гібридних підходів, що поєднують LLM із формальними методами статичного аналізу.

IRIS [13] є прикладом нейросимвольного підходу, що систематично поєднує LLM зі статичним аналізом для виявлення вразливостей на рівні репозиторію. Система використовує LLM для автоматичного виведення taint-специфікацій та контекстуального аналізу, що дозволяє зменшити залежність від ручного кодування специфікацій. На датасеті CWE-Bench-Java система IRIS із GPT-4 виявляє 55 вразливостей порівняно з 27 для традиційного аналізатора CodeQL.

LLift [14] представляє фреймворк, що використовує LLM для допомоги статичному аналізу у виявленні дефектів use-before-initialization (UBI). Система демонструє 50% точність у реальних сценаріях та ідентифікувала 13 раніше невідомих UBI-помилок у ядрі Linux. Ці результати підтверджують ефективність поєднання нейронних та символічних компонентів.

Таблиця 2. Порівняння підходів до аналізу коду

Критерій	Символічний	Нейронний	Нейросимвольний
Інтерпретованість	Висока	Низька	Середня-Висока
Масштабованість	Обмежена	Висока	Середня-Висока
Формальні гарантії	Так	Ні	Частково
Адаптивність	Низька	Висока	Висока
Хибнопозитивні	Високі	Варіативні	Знижені

Основні виклики нейросимвольних підходів включають складність інтеграції гетерогенних компонентів, обмеження масштабованості символічних компонентів та потребу в якісних бенчмарках для оцінки гібридних систем. Систематичний огляд [4] виявив, що інтерпретованість та надійність залишаються недостатньо дослідженими (28%), а метакогнітивні аспекти — найменш представленими (5%).

Перспективними напрямками розвитку є: інтеграція SMT-вирішувачів для формальної верифікації тверджень, згенерованих LLM; розробка міжпроцедурного аналізу потоків даних із використанням нейронних компонентів; створення локальних LLM, оптимізованих для задач програмного аналізу; та розробка адаптивних механізмів налаштування на основі зворотного зв'язку від користувачів.

Нейросимвольні підходи представляють перспективний напрямок, що поєднує переваги символічних та нейронних методів. Подальший розвиток потребує вирішення викликів масштабованості та розробки надійних методів верифікації гібридних систем.

Нейросимвольний штучний інтелект має значний потенціал для трансформації галузі статичного аналізу програмного коду, забезпечуючи більш ефективні, точні та інтерпретовані інструменти для забезпечення якості та безпеки програмного забезпечення.

Список використаних джерел

1. Jang-Jaccard, J., & Nepal, S. (2014). A survey of emerging threats in cybersecurity. Journal of Computer and System Sciences, 80(5), 973–993.
2. Guo, W., Xu, Y., & Li, Z. (2023). Mitigating false positives in static program analysis. ACM Computing Surveys, 55(9), 1–35.
3. Hitzler, P., Eberhart, A., Ebrahimi, M., Sarker, M. K., & Zhou, L. (2022). Neuro-symbolic approaches in artificial intelligence. National Science Review, 9(6), nwac035.
4. Colelough, B. C., & Sarker, M. K. (2025). Neuro-symbolic AI in 2024: A systematic review. arXiv preprint arXiv:2501.05435. https://doi.org/10.48550/arXiv.2501.05435
5. Bouneffouf, D., & Aggarwal, C. C. (2022). Survey on applications of neurosymbolic artificial intelligence. arXiv preprint arXiv:2209.12618. https://doi.org/10.48550/arXiv.2209.12618
6. Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nature Machine Intelligence, 1(5), 206–215.
7. Marra, G., Dumančić, S., Manhaeve, R., & De Raedt, L. (2024). From statistical relational to neurosymbolic artificial intelligence: A survey. Artificial Intelligence, 328, 104062.
8. Feng, Z., Guo, D., Tang, D., Duan, N., Feng, X., Gong, M., ... & Zhou, M. (2020). CodeBERT: A pre-trained model for programming and natural languages. Findings of EMNLP 2020, 1536–1547.
9. Guo, D., Ren, S., Lu, S., Feng, Z., Tang, D., Liu, S., ... & Zhou, M. (2021). GraphCodeBERT: Pre-training code representations with data flow. Proceedings of ICLR 2021.
10. Wang, Y., Wang, W., Joty, S., & Hoi, S. C. (2021). CodeT5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation. Proceedings of EMNLP 2021, 8696–8708.
11. Guo, D., Lu, S., Duan, N., Wang, Y., Zhou, M., & Yin, J. (2022). UniXcoder: Unified cross-modal pre-training for code representation. Proceedings of ACL 2022, 7212–7225.
12. Zhang, J., Li, Y., Chen, X., & Wang, Z. (2025). When LLMs meet cybersecurity: A systematic literature review. Cybersecurity, 8(1), 1–42. https://doi.org/10.1186/s42400-025-00361-w
13. Li, Z., Shi, B., Liu, S., Chen, J., Khurshid, S., & Naik, M. (2024). IRIS: LLM-assisted static analysis for detecting security vulnerabilities. arXiv preprint arXiv:2405.17238.
14. Li, H., Hao, Y., Zhai, Y., & Qian, Z. (2023). Enhancing static analysis for practical bug detection: An LLM-integrated approach. Proceedings of ESEC/FSE 2023.