Name of student : FAKIR Yasmine

Name of your level 1: IGUEME Merveille

Title: Combining Foundation Models and Symbolic AI for Automated Code Vulnerability Detection

Key words: Foundation Models, Symbolic AI, Code Vulnerabilities, Large Language Models (LLMs), Abstract Syntax Trees (ASTs), Software Crisis, Automated Software Factory, Static Source-code Analysis, Code Generation, Ethical Considerations, Collaborative Systems, Cloud-based Systems, Transformer Models, Machine Learning, Automatic Code Patching.

Summary:
The article discusses the integration of Foundation Models and Symbolic AI for automating the detection and mitigation of code vulnerabilities. It highlights the challenges in existing approaches, such as symbolic AI and run-time testing, and proposes a solution combining code generation using Large Language Models (LLMs) with scalable defect elimination methods using symbolic AI. The authors emphasize the need for automation in creating defect-free code, particularly in the context of collaborative and cloud-based systems.

The background section refers to the long-standing "Software Crisis" and the brittle nature of software, especially in Cyber-Physical Systems (CPSs). The introduction introduces Transformer Models as a general-purpose architecture, emphasizing their success in various domains, including code generation. It acknowledges the limitations of Software 2.0 and the potential issues with transformer models like ChatGPT and CODEX.

The related work section mentions the lack of application of Formal Methods to code generated by Foundation Models and introduces the DARPA Assured Micropatching (AMP) program. The problem definition outlines the two categories of transformer models and their deficiencies, including runtime errors in generated code. The article stresses the importance of addressing such errors for the successful implementation of Foundation Models in software development.

The authors propose an "Automated Software Factory" that involves studying the flaws in AI-generated code and using symbolic AI for analysis. The technical approach includes static source-code analysis and matching/rewriting infrastructure using Abstract Syntax Trees (ASTs). The process involves understanding and cataloging errors, designing symbolic AI tools for analysis, and experimenting with custom Foundation Models trained on legacy code.

The conclusion underscores the potential of Foundation Models in mitigating the software crisis by automating code generation and patching. It emphasizes the need to address concerns related to intellectual property, bias, accountability, and job displacement. The ethical statement acknowledges the potential benefits of Large Language Models (LLMs) but also raises awareness about responsible and ethical usage.

AI Model Used: Abstract Syntax Trees (ASTs), Symbolic AI, Static Source-code Analysis, Cyber-Physical Systems (CPSs), Software Crisis, Formal Methods, DARPA Assured Micropatching (AMP) Program, Ethical AI, Safety-Critical Systems, Software Development, Generalist System Theory.