Title

Temporal Dependency Unfolding: Improving Code Generation for Complex Stateful Systems

Problem Statement

Generating code for complex, stateful systems or applications with intricate temporal dependencies remains challenging for current code generation models. Most existing approaches focus on generating individual functions or small code snippets without fully considering the temporal aspects and state changes in larger systems. This limitation hinders the applicability of AI-assisted programming in areas such as distributed systems, game development, and real-time applications

Motivation

Many real-world applications require careful management of state over time. Existing code generation models struggle with capturing the full complexity of temporal dependencies and state changes in larger systems. A method that can effectively reason about and generate code for systems with complex temporal dependencies could significantly improve the applicability of AI-assisted programming in critical areas. Our proposed Temporal Dependency Unfolding method is inspired by how human developers approach complex system design, first identifying key states and their relationships before implementing the detailed logic.

Proposed Method

We propose Temporal Dependency Unfolding, a novel prompting technique that guides the model to generate code by explicitly reasoning about state changes and temporal relationships. The method consists of five steps:

1. State Identification: Prompt the model to identify key states and variables that change over time in the target system.
2. Temporal Graph Construction: Guide the model to create a conceptual graph of how these states evolve and interact over time.
3. Staged Code Generation: Generate code in stages, focusing on different temporal slices or state transitions in each stage.
4. Consistency Verification: After each stage, prompt the model to verify temporal consistency and make necessary adjustments.
5. Integration: Finally, guide the model to integrate the stage-wise generated code into a cohesive system, ensuring proper handling of all temporal dependencies.