AI 번역가와 프로그램 합성의 조화

한승우, KAIST 전산학부

현재 AI 코딩 도구는 '그럴듯하게 틀린' 코드를 제안해 보안 위험을 야기하며, 증명 기반의 정형 합성은 명세 작성의 어려움으로 인해 널리 쓰이지 못했다. 본 에세이는 이 둘의 조화를 해법으로 제시한다. AI를 최종 코드 작성자가 아닌, 인간의 프롬프트를 '엄밀한 명세(formal specification)'로 변환하는 '번역가'로 활용하고, 프로그램 합성 도구가 이를 '증명'하며 구현하는 하이브리드 모델이다. 이 방식은 AI의 사용성과 정형 합성의 신뢰성을 결합하여, 개발자의 역할을 '그럴듯하게 틀린' 코드의 감사자에서 '구조적으로 정확한' 시스템의 설계자로 격상시킨다.

프로그래밍의 미래에 대해 현재 '빠르고 위험함 AI'와 '느리고 안전한 프로그램 합성(program synthesis)'이라는 상반된 두 전망이 충돌하고 있다. 하나는 AI가 인간처럼 코드를 작성하며 코딩을 쉽게 할 것이라는 대중 매체의 장밋빛 예측이다 [2, 5]. 다른 하나는 AI가 개발자를 속여 그럴듯하게 틀린 코드를 수용하게 만든다는 눈앞에 닥친 위험한 현실이다 [4]. 이와 별개로, 정형 논리(formal logic)에 기반해 검증된 코드를 생성하는 미래도 연구되어 왔지만, 높은 진입 장벽 탓에 실제로 적용되거나 주목받지 못했다.

하지만 이 둘은 경쟁 관계가 아니며, 프로그램의 진정한 진화는 AI를 '번역가'로, 프로그램 합성을 '구현자'로 삼는, 둘의 조화에서 찾을 수 있다. 우리는 '빠르지만 틀린' 대규모 언어 모델과 '정확하지만 어려운' 프로그램 합성이라는 두 패러다임 사이에 갇힌 셈이다. 이 문제를 해결할 열쇠는 AI가 최종적으로 코드를 작성하는 것이 아닌, AI가 인간의 '프롬프트'를 '수학적 증명 기반' 합성 엔진이 구현할 수 있는 엄밀한 명세로 바꾸는 '번역가'의 역할을 하는 것이다.

현재 GitHub Copilot 같은 AI 코딩 도우미는 확률에 의존하여, 보안에 취약한 '그럴듯하게 틀린' 코드를 제안하여 개발자를 '자동화 편향'에 빠지게 한다는 근본적인 한계를 지닌다. 이 도구들은 방대한 공개 코드를 학습해 가장 흔한 패턴을 흉내 낼 뿐이며, 이 패턴에는 SQL 인젝션 같은 심각한 취약점이 가득하다 [4]. Microsoft Excel이 유전자 이름을 날짜로 자동 변환해 수십 년간 과학 데이터를 오염시킨 오류[3]처럼, 이 AI의 제안은 도움이 되도록 설계되었기에 더 교활하고 위험하다. 이러한 '자동화 편향'에 빠진 개발자는 AI가 제안한 그럴듯하게 틀린 코드를 비판 없이 수용하기 쉽다.

반면, 이 문제에 대한 철학적 해독제로 여겨지는 정형 프로그램 합성은 '정확성을 증명 가능한' 코드를 만들 수 있지만, 개발자가 일반적인 코드보다 어려운 '엄밀한 명세'를 작성해야 한다는 치명적인 병목이 있다. 이 접근 방식은 통계가 아닌 정형화된 검색 방법에 기반하여 정확한 코드만을 생성한다 [1]. 이 방식이 AI의 신뢰 문제를 해결할 수는 있지만, 완벽한 엄밀한 명세를 요구하는 특성 때문에 그 강력함에도 불구하고 아직 틈새 기술(niche technology)에 머물러 있다.

진정한 해법은 AI의 자연어 이해력을 이용해 이 '명세 병목 현상'을 해결하는 하이브리드 모델에 있다. 이 모델이 제시하는 새로운 작업 흐름은 다음과 같다. 첫째, 개발자가 "사용자 이름을 받아 프로필을 반환하는 안전한 DB 쿼리 함수"처럼 자연 언어로 프롬프트를 작성한다. 둘째, AI 모델이 '번역가'가 되어 이 프롬프트를 코드가 아닌 엄밀한 명세로 변환한다. 셋째, '증명' 기반의 합성 엔진이 이 명세를 받아, 제약 조건을 수학적으로 만족하는 코드를 구축한다.

이 하이브리드 모델은 오류를 내기 쉬운 인간과 결함 있는 AI 모두를 최종 코드 작성 단계에서 배제함으로써, '자동화 편향'의 위험을 원천적으로 차단한다. AI의 통계적 추측은 논리적이고 검증 가능한 합성 엔진의 엄격한 통제를 받게 되기 때문에 우리는 AI의 사용성과 정형 합성의 신뢰성이라는 두 마리 토끼를 모두 잡을 수 있다.

결과적으로 이 조화는 '그럴듯하게 틀린' 코드를 감사하는 AI의 파트너가 아닌, '구조적으로 정확한' 시스템을 지휘하는 고차원적 설계자로서 개발자의 역할을 근본적으로 격상시킨다 [5]. 우리는 마침내 단순한 '자동 수정'에서 '구조적으로 정확한 자동 생성'으로 나아가게 된다. 이것이 바로 AI가 코딩을 진정으로 '쉽게 하는' 방식이다. 인간을 대체하는 것이 아니라, 마침내 우리의 신뢰를 받을 만한 시스템을 구축하도록 돕는 방식으로 말이다.

References

[1] Rajeev Alur, Rishabh Singh, Dana Fisman, and Armando Solar-Lezama. 2018. Search-based program synthesis. *Commun. ACM* 61, 12 (Nov. 2018), 84–93. doi:10.1145/3208071

[2] Douglas Heaven. 2021. AI Can Write Code Like Humans—Bugs and All. *WIRED* (2021). https://www.wired.com/story/ai-write-code-like-humans-bugs/

[3] Dyani Lewis. 2021. Autocorrect Errors in Excel. *Nature* (2021). https://www.nature.com/articles/d41586-021-02211-4

[4] Hammond Pearce, Baleegh Ahmad, Benjamin Tan, Brendan Dolan-Gavitt, and Ramesh Karri. 2025. Asleep at the Keyboard? Assessing the Security of GitHub Copilot's Code Contributions. *Commun. ACM* 68, 2 (Jan. 2025), 96–105. doi:10.1145/3610721

[5] Craig S. Smith. 2022. Coding Made AI—Now, How Will AI Unmake Coding? *IEEE Spectrum* (2022). https://spectrum.ieee.org/ai-code-generation-language-models