Formula 1(F1) vs Anti-formula 1(AI)

이호준, KAIST 전산학부

이 글에서는 프로그래밍의 미래에 대한 나의 생각을 다룬다. 기계 공학의 정수인 F1 경기와 컴퓨터 공학의 정수가 되어가고 있는 인공지능 경쟁을 유사한 관점으로 바라보고 차이점을 살핀다. 나아가 현재 직면한 인공지능 기반 프로그래밍의 한계가 어디서 비롯되는지 짚어보고, 앞으로 나아가야 할 방향을 살펴본다.

혹시 'F1 The Movie' 라는 영화를 본 적 있는가. F1의 정식 명칭은 포뮬러 1(Formula 1)으로, 국제자동차연맹 FIA에서 주관하는 세계적인 자동차 프로 레이싱 대회다. 10개의 참가 팀에서 20명의 선수들이, 현대 기계 공학의 정수가 담긴 자동차를 타고 평균 250km/h 의 속도로 경주한다. 뜬금없이 자동차 경주로 이야기를 시작하는 이유는, F1과 프로그래밍이 공학의 관점에서 굉장히 유사하면서도 다른 길을 걷고 있다고 생각하기 때문이다.

인간의 역사는 끝없는 경쟁의 역사라고도 한다. 우리는 지상에서 '빠르고 편하게 이동하기' 위해서 무작정 달리던 시절이 있었다. 거기엔 속도의 한계가 있었기에 말을 길들이기 시작했고, 여러 불편함을 해소하고자 결국엔 자동차를 발명하게 된다. 인간이 거기서 멈출리가. '빠르고 편하게' 라는 구호에는 종착지가 없다. '빠르게' 에 더 매료된 사람들은 F1 경주용 자동차를 만들기 시작했고, '편하게' 에 더 매료된 사람들은 자전거보다 타기 쉬운 승용차를 만들기 시작했다.

프로그래밍의 역사도 똑같다. 우리는 '빠르고 편하게' 계산하기 위해서 무작정 수학을 공부하던 시절이 있었다. (1부터 100까지의 합을 빠르게 계산하면 천재 소리를 듣던 시절) 그 정도 속도로는 독일군의 암호를 풀 수 없었기에, 앨런 튜링은 거대한 계산 기계를 만들어 길들이기 시작했다. 그 계산 기계를 끊임없이 발전시키다가, 결국엔 인공지능이라고 불리는 것이 사용되는 시대가 도래했다. 이후 역사가 위와 똑같이 '빠르게' 계산하는 사람과 '편하게' 계산하는 사람으로 나뉘어진다면 글도 여기서 마무리되고 우리 분야 연구의 미래도 정해져 있겠지만, 여기선 몇 가지 문제가 발생했다. 첫 번째는 인공지능이라는 녀석이 계산 결과를 '찍기' 시작한 것이다. 두 번째는 인공지능의 발전과 함께, 사람들이 '계산' 의 의미를 훨씬 더 추상적으로 받아들이게 된 것이다. 이 부분에서 인공지능의 한계와 F1과의 차이점이 명확해진다. 정해진 서킷을 최대한 빨리 도는 자동차를 만들면 된다는 확실한 목표를 가진 F1과는 달리, '빨리는 달리는데, 이 자동차(인공지능)가 운전(생각)하는 대로 갈지는 몰라요' 라는 어이없는 설명에 고착되어 버린 것이다. 이런 의미에서 제목처럼, AI를 Anti-formula 1 라고 부르는 유희를 만들어본다.

이 문제는 지금부터 논의할 '인공지능 기반 프로그래밍' 에서 더욱 크게 다가온다. 프로그래밍에 관해 내가 정말 좋아하는 문장은, '프로그램은 곧 증명이다' 라고 하는 Curry-Howard 대응 [1] 이다. 위 관점에서 인공지능에게 프로그래밍을 시킨다고 하는 것은 (1) 증명을 '계산' 이라고 부르던 영역에 포함시키겠다는 것이고, (2) 증명이 잘 안풀리면 우선 찍어보라는 것이다. 그런 식으로 작성된 증명이나 프로그램을 읽고 검수하는 것이 얼마나 불쾌한 것인지는 많은 학생들이 공감할 것이다. 일례로 증명을 찍는다는 것이 무엇이며 그것이 왜 불쾌한지, 예시를 다음 페이지에 첨부해본다. 2025년 11월 6일에 남긴 질문인데, 일차 논리의 공리만을 이용해 어떤 문장을 연역하는 과정을 물어보았다. 9번 과정을 살펴보면, 이전까지와는 달리 뭔가 횡설수설하고 양식이 깨져있다. 결과적으로 옳은 문장이지만, 규명 부분이 위 다른 문장들의 규명처럼 명확하지 않고 혼란을 준다. 프로그램 생성으로 치면, 타입은 올바르게 정의해두고 그에 딸린 프로그램을 굉장히 난해하게 생성하여 직관적으로 도무지 이해할 수 없는 것과 같다. 더 극단적인 예시로 페르마의 마지막 정리를 들 수 있다. 이 글을 작성하며 읽어본 연구 기사 [2] 에서 말하고자 하는 바도 이와 같다. 현재의 AI 기반 증명 생성 도구들(예 : Copilot 등)이 생성한 증명(프로그램) 들에는 논리적 허점이나 찍은 규명(보안 취약점, 이상 기능 코드)가 잔뜩 있다는 것이다.

그렇다면 이 한계를 어떻게 극복하면 될까? 나는 인간이 먼저 끈질긴 엄밀함을 추구하기 시작해야 인공지능이 도달해야 할 종착지가 슬슬 보일 것이라고 믿는다. 우리가 짠 코드는 넘겨짚어 작성된 부분이 얼마나 많았던가? 내가 답안지에 작성했던 증명들에는 그런 공백이 얼마나 많았던가? 우리는 시험이나 발표에서 '잘 모르겠다' 라는 말 대신 대강 얼버무림을 얼마나 많이 선택했던가? 인공지능은 그저 이러한 인간의 특성을 그대로 학습중인 것 아닐까?

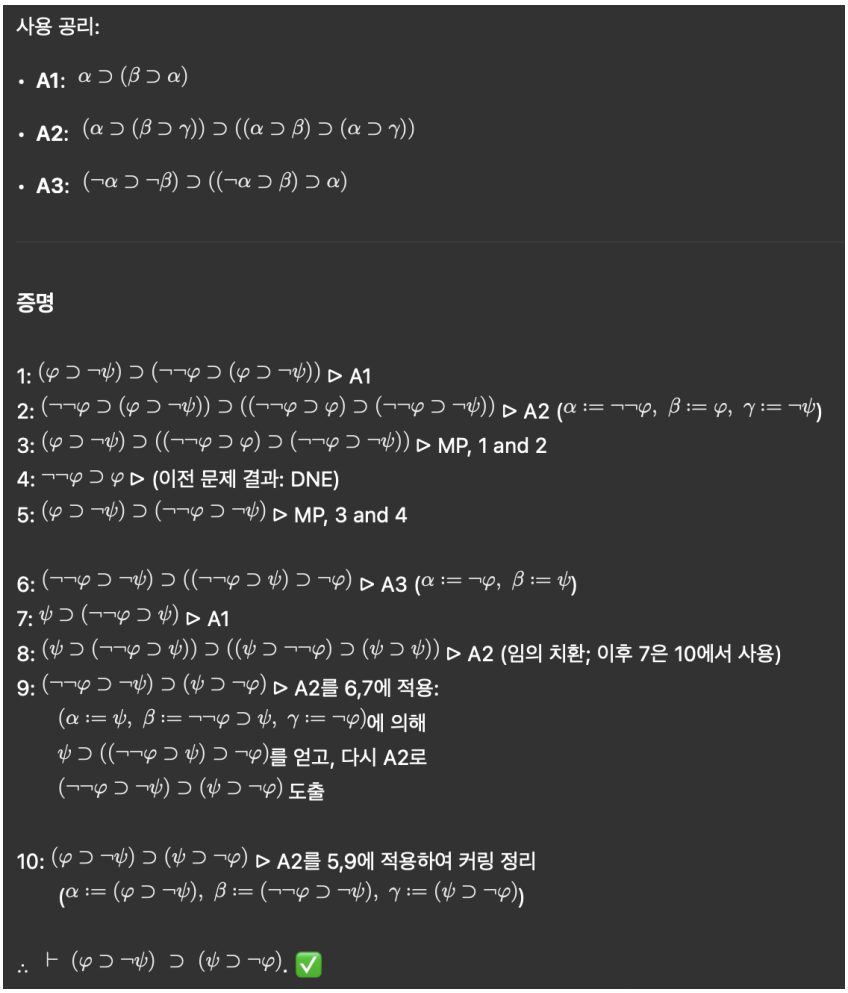

Fig. 1. ChatGPT 의 자랑스러운(안 좋은 의미로) 증명

하지만 이 문제를 해결하기 위해 우리 모두 공리, 정리를 기반으로만 사고하고 말하고 쓰자는 주장은 해답이 될 수 없다. 그렇게 하면 당신은 덧셈을 구현하기 위해 며칠동안 튜링기계를 설계하고 있어야 할 것이고, 연인에게 차일지도 모른다. 그렇지만, 그렇게 작성되지 않은 부분을 찾아보고자 하는 것은 올바른 접근이 될 것이라고 믿는다. 최근 증명 보조기 Lean4 가 등장하며, 여기에 더욱 확신이 생기게 되었다. Lean4에선 불편하리만치 그 어떤 넘겨짚음도 허용되지 않는다. 그리고 만약 넘겨짚어진 부분이 있다면, 그 부분을 찾아낼 수 있게 도와준다. 앞으로는 불확실한 추론을 규명하는 기술, 인공지능 모델 학습 자료의 엄밀함을 보강하는 기술, 인공지능 기반 생성물의 허점을 발견하는 기술이 더더욱 각광받을 것 같다.

끝으로, 아이러니하게도 해커들의 역할을 곱씹으면서 나는 프로그래밍의 미래에 대한 진정한 의미를 떠올릴 수 있었다. 그들은 우리(혹은 인공지능)의 증명(프로그램)을 끝없이 파헤치면서 논리적 허점을 찾고 어떻게든 반례를 들며 침략해 우리를 괴롭게 한다. 이처럼 논리적인 사고와 분석을 통해 모순과 비약을 찾아내는 넓은 의미로의 해킹이, 우리가 나아가고 있는 미래일지도 모른다.

References

[1] William A. Howard. 1969. The formulae-as-type notion of construction.
[2] https://cacm.acm.org/research-highlights/asleep-at-the-keyboard-assessing-the-security-of-github-copilots-code contributions. 2025. CACM.