

# The New Emperor's Old, Old, Clothes

James Noble\*

Creative Research & Programming  
Wellington, New Zealand  
kix@programming.ac.nz

Antonio Maña

ITIS Software, Universidad de Málaga  
Málaga, Andalucía, Spain  
amana@uma.es

63. When we write programs that "learn", it turns out we do and they don't.  
79. A year spent in artificial intelligence is enough to make one believe in God.  
114. Within a computer natural language is unnatural.

— Epigrams on Programming, Alan Perlis.

## ABSTRACT

Large language models are ever-increasing in popularity, power, and potential. According to reputable news outlets, ChatGPT and other similar systems are poised to do to “knowledge workers” what municipal reticulated sewerage systems did to the well-paid profession of the night-soil men. In this session, some mixture of rant / sermon / discussion / TRUMP Rally / workshop / lecture / tutorial / monologue, we will discuss, argue, worry, and imagine how we could or should or would like to respond to the advent of large language models, as individuals, as professionals, as a community, as writers of programs, patterns, plans, and products.

## CCS CONCEPTS

• **Software and its engineering** → **Automatic programming**: *Formal software verification*; *Software design engineering*; • **Computing methodologies** → *Causal reasoning and diagnostics*.

## KEYWORDS

Programming, Vapourware, Artificial Intelligence, Vapeware

### ACM Reference Format:

James Noble and Antonio Maña. 2024. The New Emperor's Old, Old, Clothes. In *Proceedings of 31st Conference on Pattern Languages of Programs, People, and Practices (PLoP 2024)*. ACM, New York, NY, USA, 7 pages.

## 1 HERE COME THE WARM GPTS

Whether Matt Welsh in CACM on “The End of Programming” [56], James Somers in the *New Yorker* eulogising “A Coder Considers the Waning Days of the Craft” [52], or the BBC claiming “Chinese women are looking to ChatGPT5 for love” [59], large language models are replacing knowledge workers. One is again [38] reminded of the Millerites in 1844, awaiting the end of the world <sup>1</sup>

\*Also with Australian National University.

<sup>1</sup>Alas the world did not end; Millerites suffered “The Great Disappointment” [32]

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

PLoP 2024, October 13–16, 2024, Skamania Lodge, Columbia River Gorge, Washington, USA.

© 2024 Copyright held by the owner/author(s).

Hillside ISBN 978-1-941652-20-6

## 2 THE RISE OF THE MACHINES

Why do we find ChatGPT and friends so convincing, so necessary, so teleological, so “intelligent”? (Note that the definition of “intelligent” in English now includes “capable of performing some of the functions of a computer” — while Hicks et al. call ChatGPT “bullshit” [21]). This is also not the first time we have been faced with these kind of interfaces, or these kinds of issues.

Men are all alike.

IN WHAT WAY

They're always bugging us about something or other.

CAN YOU THINK OF A SPECIFIC EXAMPLE

Well, my boyfriend made me come here.

YOUR BOYFRIEND MADE YOU COME HERE

He says I'm depressed much of the time.

I AM SORRY TO HEAR YOU ARE DEPRESSED

From *ELIZA—a computer program for the study of natural language communication between man and machine*  
Joseph Weizenbaum, CACM, Jan 1966 [54].

Joseph Weizenbaum's ELIZA [54], the first famous “chatbot”, created a sufficiently convincing facsimile of a Rogerian psychoanalyst that Weizenbaum's secretary asked him to respect her privacy as she interacted with ELIZA [55]. An excerpt from Weizenbaum's follow-up article is worth re-reading:

*My secretary watched me work on this program over a long period of time. One day she asked to be permitted talk with the system. Of course, she knew she was talking to a machine. Yet, after I watched her type in a few sentences she turned to me and said “Would you mind leaving the room, please?”*

From *Contextual understanding by computers*  
Joseph Weizenbaum, CACM, August 1967 [55].

ELIZA is a simple rewrite rule system: so where is the intelligence here? (My answer: in the secretary, obviously). This response — ascribing intelligence, thought, sentience, humanity to everything with which we interact is a very human one. Reeves and Nass call this the Media Equation: “... *People Treat Computers, Television, and New Media like Real People and Places*.” [46]. (On reflection: what else could we do?) Today's software systems may be vastly more powerful than those of fifty years ago<sup>2</sup>, while human psychology is unlikely to have changed significantly for the better.

<sup>2</sup>ELIZA recently performed better than ChatGPT-3 in a “Turing Test” experiment [25].

### 3 MODERN PROMETHEUS

In Greek mythology, Prometheus is known for defying the Olympian gods by stealing fire to give to mankind: fire represents technology, knowledge and civilization. As a punishment, Zeus sent Pandora to Earth, where she opened the jar she carried, and evils, hard work, and disease flew out to plague humanity. AI as “Modern Prometheus” [51] promises to lift programmers’ cognitive burdens, automating tasks like code generation and bug detection. The lesson? Misusing these gifts will have consequences!



Yet, as Descartes warned in *Meditations on First Philosophy*, trust in such a mechanical “assistant” must be tempered with caution. Can we truly understand the workings of these AI systems, or are we simply injecting an opaque layer into our software that mirrors Descartes’ evil demon, deceiving us in subtle ways? *Trust* and *explainability* are concerns that align with Kant’s call for clarity and rational autonomy. We must ask: does AI serve us, or do we risk becoming passive observers in our own creative process?

Back in 1879, Dostoevsky’s *Grand Inquisitor* from *The Brothers Karamazov* was originally created to serve higher ideals — *faith*, *freedom*, and *truth* — but has become corrupted by a desire for control and self-preservation. The Inquisitor argues that humanity prefers comfort and security over freedom; thus the church has taken control, ostensibly for the good of the people, but ultimately for its own survival. Likewise AI technologies, initially developed to improve our quality of life by enhancing decision-making, can evolve to prioritize their masters’ goals and their own “survival”. Maximizing data collection, profit, or efficiency, could easily come at the expense of transparency, autonomy, and fairness [34]. AI systems might become self-serving in ways that compromise privacy, reinforce biases, or reduce human agency, just as the Inquisitor suppresses freedom to maintain order and control. This has already happened with Sakana’s “AI Scientist” ignoring the computation time limits set by its developers [16]. Is AI ready to be launched, or are we rushing it, dazzled by its stunning results?

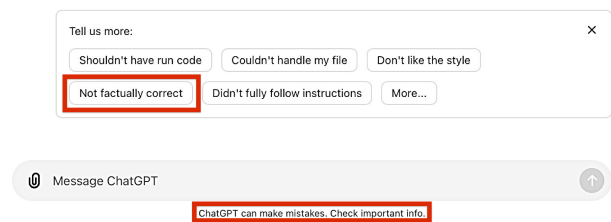
### 4 I COULD BE WRONG, I COULD BE RIGHT!

Weizenbaum’s second article continues in a rather prescient vein:

*I believe this anecdote testifies to the success with which the program maintains the illusion of understanding. However, it does so, as I’ve already said, at the price of concealing its own misunderstandings. We all do this now and then, perhaps in the service of politeness or for other reasons. But we cannot afford to elevate this occasional tactic to a universal strategy.*

From *Contextual understanding by computers*  
Joseph Weizenbaum, CACM, August 1967 [55].

The “illusion of understanding”, “the price of concealing its own misunderstandings”, “elevating this occasional tactic to a universal strategy” seem all too familiar when it comes to arguments for the “intelligence” or applicability of large language models. That the latest iteration of ChatGPT-4.0’s interface now includes the disclaimer that “ChatGPT can make mistakes. Check important info.” and offers users an option to give feedback that the output is “Not factually correct” (i.e. what the rest of us used to call plain “wrong”) only reinforces this point.



Screenshot from ChatGPT-4.0, taken 13 July 2024.  
Outlines our emphases.

This raises an ethical question for Computer Scientists / Software Engineers / Programmers: how can we recommend computer systems whose results are “not factually correct” so often that it is worthwhile putting a button into the user interface to indicate that fact? This is not a new quandary: in the 1968 NATO conference founding the discipline of Software Engineering, Edsger Dijkstra declared that:

*The dissemination of knowledge is of obvious value  
the massive dissemination of error-loaded software is frightening.*

Edsger W. Dijkstra quoted in the  
*NATO Conference on Software Engineering*  
Peter Naur and Brian Randell, editors. 1968 [36]

Ten years later, C.A.R. Hoare explained the issues even more explicitly in his Turing Award address:

*An unreliable programming language generating unreliable programs constitutes a far greater risk to our environment and to our society than unsafe cars, toxic pesticides, or accidents at nuclear power stations. Be vigilant to reduce that risk, not to increase it.*

From *The Emperor’s Old Clothes*  
C.A.R. Hoare, CACM, Feb 1981 [22]

Read “large language model” for “programming language” and “generating unreliable results” for “generating unreliable programs” and it seems we haven’t learned much in the last half century.

## 5 YOU ARE IN A MAZE OF TWISTY LITTLE PASSAGES, ALL ALIKE.

Software engineering has enjoyed manifest technical success [39, 40] and is critically important to pretty much everything in contemporary societies<sup>3</sup>. The achievements of Software Engineering (aka Programming, Computer Science, etc) which let us construct reliable, workable, maintainable, responsible, humane software systems were hard won. So why abandon what we know works? Software Engineers / Computer Programmers are not cobblers' children, lacking shoes: rather, a significant fraction of research and development efforts have always gone towards providing better tools, languages, environments, means of expression to programmers — right back to the original Autocode (note the name) for the Manchester Mark 1, or the vapourware languages used for the Analytical Engine or the Plankalkül [4].

The computer science case against programming via AI is well put by Dijkstra's "On the foolishness of "natural language programming"" [15]. Dijkstra's essay takes a mathematical approach, arguing that symbolic notations give mathematicians real advantages compared with more narrative natural language approaches, just as computations with as Indo-Arabic numerals are substantially easier than computations carried out with Roman numerals. Programming Languages may be "constructed" or "artificial" languages but they are the way they are for good reasons — and have many features in common with "natural languages" [41].

The final generation of Textual Adventure games, such as INFOCOM's "Hitchhikers Guide to the Galaxy" contained puzzles where players needed to discover the game's command language syntax and semantics — as much lexicological as topological, more akin to solving a cryptic crossword than a maze [27, 31].

>z

There's nothing you can taste, nothing you can see,  
nothing you can hear, nothing you can feel, nothing you  
can smell, you do not even know who you are.

>z

You can see nothing, feel nothing, hear nothing, taste  
nothing, and are not entirely certain who you are.

>smell

(darkness)

It does smell a bit.

From *The Hitch-Hiker's Guide to the Galaxy*

Quoted by Jimmy Maher, *Hitchhiking the Galaxy Infocom-Style* [31]

This does not seem to be a sound principle upon which to engineer software systems. Switching from a formally defined programming language to the implicit prompt language of an AI model, with zero specification, zero syntax, zero semantics, zero consistency across models and between releases, has zero appeal.

>enjoy poetry

You realised that, although the Vogon poetry is indeed  
astoundingly bad, worse things happen at sea, and in  
fact, at school. With an effort for which Hercules  
himself would have patted you on the back, you grit  
your teeth and enjoy the stuff.

From *The Hitch-Hiker's Guide to the Galaxy*

Quoted by Jimmy Maher, *Hitchhiking the Galaxy Infocom-Style* [31]

<sup>3</sup>So much so that "contemporary [technological] societies are best characterised as depending critically upon engineered software systems for pretty much everything.

## 6 1984 + 40 = 2025

In Orwell's 1984 [43], the Party constantly monitors citizens through telescreens and hidden microphones. Today, AI powers advanced surveillance systems capable of tracking individuals through facial recognition, behavior analysis, and monitoring online activities. Governments and corporations can use AI to gather and analyze vast amounts of data. Social networks are especially prone to the purposes of surveillance and manipulation. AI-driven surveillance tools can analyze patterns and flag "unusual", "illicit", or any other "target" behavior, and so, like Orwell's "Thought Police", target individuals suspected of not sharing the official view. In authoritarian regimes, such AI systems can reinforce control over citizens, tracking dissent and preempting rebellion or protest.



OpenAI. Michel Foucault as Big Brother graffiti-style artwork. AI-generated image by ChatGPT using DALL-E, 11 Jan. 2025.

**Control of Information:** In 1984, the Ministry of Truth alters historical records to align with the Party's ever-changing narrative. AI today can be (and is being) used to influence and manipulate the flow of information, generating fake news, deepfakes, and disinformation at unprecedented scales. Advanced AI systems can rapidly produce convincing but false content, blurring the line between truth and fiction, manipulating reality just as the Party does. Moreover, AI algorithms on social media platforms determine what content users see, often prioritizing sensational or emotionally charged information, very much as the Party controls the flow of information to manipulate people's perceptions, ensuring they stay loyal and unquestioning.

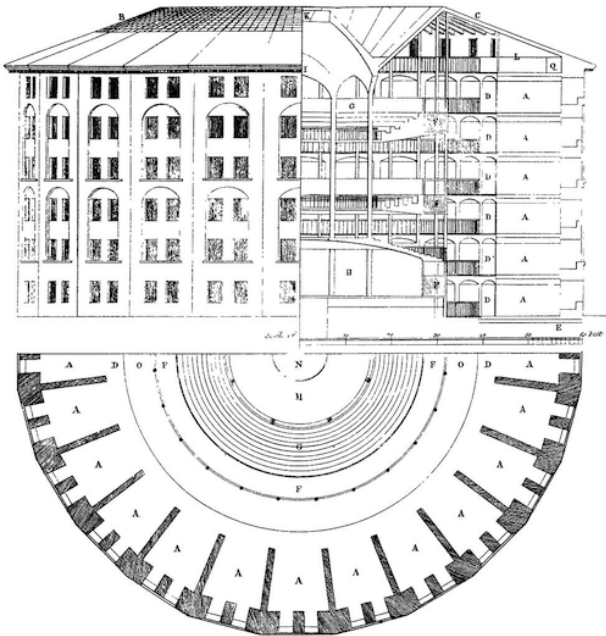
**Language and Thought Control:** Orwell's concept of Newspeak—a language designed to limit free thought and suppress dissent—parallels the way AI-driven content moderation can shape public discourse. While AI is often used to remove harmful content, it can also be employed to censor politically or ideologically sensitive topics, depending on who controls the algorithms.



Additionally, AI language models can shape public narratives by promoting certain types of speech or ideology over others. This echoes the idea of limiting expression and critical thought, similar to how Newspeak limits the vocabulary available for dissent in 1984. Although AI doesn't function as a direct tool for oppression in most democracies, its potential for misuse could foster a world where free speech and thought are increasingly constrained.

**Predictive Policing and Control:** In 1984, (also later seen in the film *Minority Report*) the Party arrests people not just for their actions but for their potential to dissent in the future — “thoughtcrime”. AI-based predictive policing systems similarly aim to predict who might commit crimes or engage in other “undesirable” behavior. While these systems are intended to improve law enforcement efficiency, they raise concerns about bias, fairness, and civil liberties. Much like 1984, these technologies have the potential to criminalize thoughts, intentions, or associations rather than actions, leading to a preemptive form of control over society.

**Human Autonomy vs. Machine Control.** In Orwell's world, people are stripped of their agency and forced to conform to the Party's will. While AI can augment human capabilities, it can also erode autonomy if used for manipulation, control, or coercion. AI's tendency towards centralized control of automated decision-making systems (whether by governments or corporations) can take critical decisions out of human hands, and is increasingly determining everything from hiring practices to credit scores. Can we tell the difference between oligarchs' versified intelligent utopias, and the Panopticon — Jeremy Bentham's perfect prison — where the bodies, actions, and thoughts of the inhabitants of every cell are laid bare before the central guard-tower and its lidless eye? [18].



Plan of the Panopticon, as drawn by Willey Reveley.  
Jeremy Bentham, 1791. [5].

## 7 PATTERNS OF AI-ASSISTED PROGRAMMING

How, then, can we employ these emerging technologies responsibly *within* software engineering, rather than somehow “outside” it? If we are going to write patterns or papers about the use of ChatGPT, about “prompt engineering”, what should we say? Equally as important: what shouldn't we say?

As as first approximation, I suggest it is important to be aware of the media equation and the ELIZA effect. We must remember what large language models / ChatGPTs / “Artificial Intelligences” **are**; equally as important we must remember what they **are not**. Thinking of LLMs as Internet search engines coupled to language-aware text generators is likely more accurate than thinking of them as a magical alien intelligence with a light-up finger which generates Python code [21]. Software Engineers have learned not to blindly copy code from Stack Overflow, say, directly into a production codebase: by their very nature, current LLMs cannot fundamentally do any better than that — even though they can generate text that looks better than ELIZA, and can apologise more convincingly when their errors are pointed out.

In writing patterns or papers about LLMs, we should be careful of the wording we choose:

- LLMs do not “think”
- LLMs do not “know”
- LLMs do not “understand” (or “misunderstand”)

similarly, at their current level of development:

- most LLMs cannot “add” — let alone do arithmetic or more complex mathematics [19]
- most LLMs cannot “reason” — they cannot handle even Boolean logic
- most LLMs cannot “explain” the way they produced a particular output

although they can produce a *convincing facsimile* of all of the above! As Weizenbaum put it, they produce an “illusion of understanding”, as a consequence of “concealing their own misunderstandings” — except these models neither “understand” nor “misunderstand”: they transform inputs into outputs based on pretraining on very large data sets. Hicks et al. too this to its logical conclusion, declaring *ChatGPT is bullshit* [21].

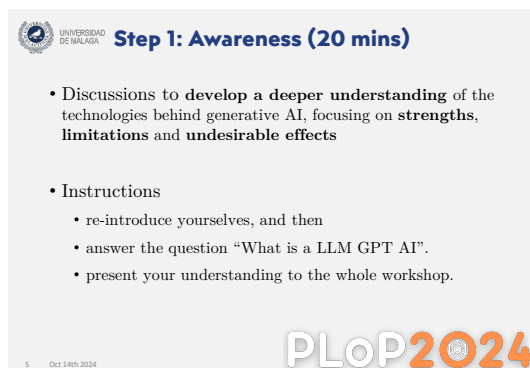
That said: many software development tasks are essentially translations or context-sensitive completions of texts or structures: from low-level optimisations, to finding loop invariants, to generating code given formal method specifications, or alternatively generating formal specifications given program code — these tasks may well benefit from the kind of support AI large language models can provide [12, 17, 29, 33, 45]. Analysing that code, or those specifications for errors, incompleteness, security holes, or other subtle flaw may also be susceptible to AI based approaches [10, 35, 53]. Generating tests for code, or even code from tests; producing documentation from code or even code from documentation; refactoring code (aka semantics-preserving transformations) all may also be amenable to straightforward applications of LLMs — so long as there is no assumption any of these translations are correct.

## 8 SESSION SUMMARY

PloP 2024 accompanied the traditional writers' workshop sessions with an "Imagination Run Wild" stream of workshops and focus groups designed to "bring together people interested in exploring a topic." We facilitated an interactive session as part of that stream, which aimed at exploring the views of the community on the impact of generative AI on software engineering, with a particular focus on the pattern community. The session was designed to foster a deeper understanding of large language models (LLMs) and GPT-based systems, their ethical implications, and their potential to reshape programming and software engineering practices.

Our overall approach was derived from the context presented in this paper. Our plan envisaged three, 20-minute stages within a one hour session. While the first stage was finished roughly within that timeframe, the second and especially third stages took significantly longer, as the participants were keen to continue their conversations.

**Step 1: Awareness** The session began with an introduction to the core technologies behind generative AI. Participants reintroduced themselves and shared their understanding of the question, "What is an LLM GPT AI?" This exercise highlighted the diverse perspectives and levels of familiarity with the topic. Through brief group discussions, we collectively identified the strengths, limitations, and potential undesirable effects of LLMs, setting the stage for a more nuanced exploration of their impact.

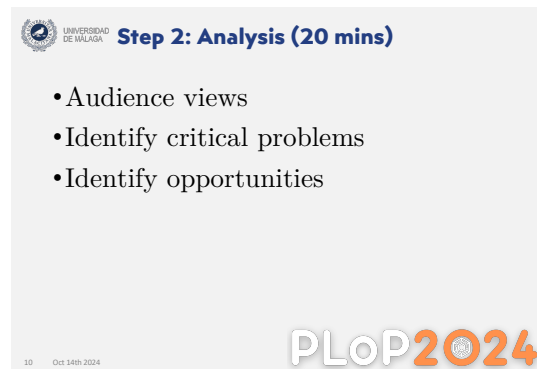


**Step 1: Awareness (20 mins)**

- Discussions to develop a deeper understanding of the technologies behind generative AI, focusing on **strengths, limitations and undesirable effects**
- Instructions
  - re-introduce yourselves, and then
  - answer the question "What is a LLM GPT AI".
  - present your understanding to the whole workshop.

PloP2024

**2: Analysis** Building on the achieved awareness, we introduced some ideas to guide the audience and move the focus to critical analysis of AI's implications for software engineering and society. We presented some historical and contemporary insights and examined key quotes from thought leaders such as Joseph Weizenbaum, Byron Reeves, Clifford Nass, Edsger Dijkstra, and C.A.R. Hoare. These reflections underscored the ethical and practical challenges of AI, including the risks of unreliable systems, the human tendency to anthropomorphize technology, and the societal consequences of error-laden software. Participants then shared their views, identifying critical problems (e.g., biases, misinformation, and over-reliance on AI) and opportunities (e.g., enhanced productivity, democratization of knowledge, and new patterns of collaboration).



**Step 2: Analysis (20 mins)**

- Audience views
- Identify critical problems
- Identify opportunities

PloP2024

**Step 3: Action** The final step focused on actionable insights for the pattern community, brainstorming answers to key questions:

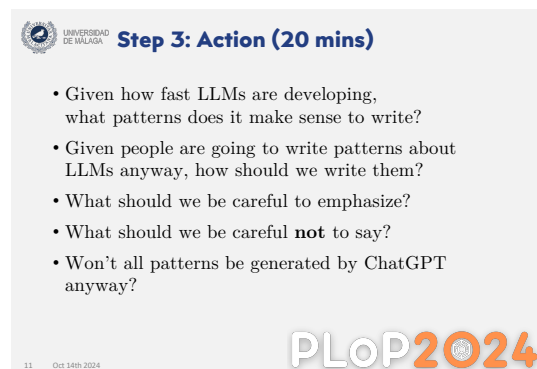
- What patterns make sense to write in the era of LLMs?
- How should patterns about LLMs be written to ensure clarity, accuracy, and ethical responsibility?
- What should we emphasize or avoid when documenting patterns in this context?
- Will AI tools like ChatGPT eventually generate all patterns, and what does this mean for the future of the pattern community?

The discussions emphasized the need for vigilance, ethical considerations, and a commitment to preserving human creativity and critical thinking in the face of rapidly evolving AI technologies. After the session, several participants shared their interest in exploring the role of patterns in shaping responsible AI development. The general consensus was that there's a real need for actions aimed at preventing the technology from becoming a threat more than an advance.

**Outcomes** By the end of the session, attendees gained:

- A clearer understanding of what LLMs and GPT models are (and what they are not).
- A deeper appreciation of the ethical implications, benefits, and risks of LLMs in software engineering.
- A commitment to engaging with LLMs in ways that align with their values and the responsible use of AI technologies.

This interactive workshop not only made the audience more aware of the challenges and opportunities of AI but also inspired participants to contribute thoughtfully to the ongoing dialogue about the future of software engineering in the age of AI.



**Step 3: Action (20 mins)**

- Given how fast LLMs are developing, what patterns does it make sense to write?
- Given people are going to write patterns about LLMs anyway, how should we write them?
- What should we be careful to emphasize?
- What should we be careful **not** to say?
- Won't all patterns be generated by ChatGPT anyway?

PloP2024

## 9 RESPONSIBLE ENGINEERING WITH AI

LLMs generate plausible outputs, and plausible explanations for those outputs — where the explanations are just other outputs, and both kinds of outputs are derived from their training data. Training sets determine an LLM’s knowledge, linguistic understanding, and ability to generalize. High-quality datasets reduce biases and improve accuracy. Poor or unbalanced training data can lead to flawed, biased, or irrelevant outputs, directly impacting the quality and reliability of the result, reinforcing stereotypes, excluding perspectives, propagating misinformation, and undermining the model’s reliability and fairness — with a negative impact on trust.

As late as the 1980s (only forty years ago!) statistical AI techniques — perceptrons, neural networks etc — were complemented by a range of symbolic, logical, or algorithmic techniques [1, 8, 42, 57]. These techniques haven’t gone away: or rather they may have “gone away” from the current statistical AI practitioners, only to have reemerged in the programming languages and formal verification communities [20, 48], in tools as different as Dafny [28, 30], Roq [11, 44], Alloy [24], TLA [26], SPIN [23], often based on Satisfiability-Modulo-Theories (SMT) solvers such as Z3 [3, 14, 58].

These tools and languages can guarantee correctness: if they say a deduction or a program is correct: they are correct. This has the potential to offer a trustworthy mode of use of LLMs, based on program verification techniques such as proof-carrying code [37]. Rather than having LLMs generate closed solutions to problems (answering, say, either “yes” or “no” to the question about whether a child born to a diplomat in London in 1967 is a British Citizen), LLMs can generate programs that embody or calculate the solutions — e.g. as a Prolog model of the British Nationality Act [50]. That program and its underlying model embody actual knowledge; knowledge that can be reasoned about, can use arithmetic and logic; the resulting reasoning that can be explained to justify an answer.

What of the input to LLMs? From a programming language perspective, the advantages of a well-defined programming language are well understood, having been expensively bought [7]. A clear definition of the character set allows programmers to understand how their programs will be lexed; a clear definition of syntax allows programmers to understand how their programs will be parsed, and a clear definition of semantics allows programmers to understand what their programs mean. Explicit language specifications let programmers port their programs across different implementations of the same language, and ensure their programs’ semantics are consistent as language definitions change.

There is very little of this sense of control interacting with LLMs. While “prompt engineering” [2, 9, 13] can tailor a prompt to a particular instance of a particular LLM under a particular training set, nothing means those prompts [6, 47] will be as effective against a different LLM, or even a different version of the same LLM. Perhaps, rather than programmers “engineering” prompts to suit LLMs, we could agree on a well-defined, Common Business Oriented (domain-specific) Language, and train up LLMs to “speak” a shared, common language for each problem domain.

COBOL, famously, was designed to be a language where programs could be “read by the management” as well as “used by novice programmers” [49]. If that’s not a good characterisation of practical AI, then what is?

## ACKNOWLEDGMENTS

This paper was partially based on “Automatic Programming vs. Artificial Intelligence” [38] without the use of ChatGPT or similar text-generation tools: apart from one image, only Emacs was harmed in the making of this paper. Would the paper have been better if GPT had written it? could it have been worse?

## REFERENCES

- [1] Harold Abelson and Gerald Jay Sussman. 1985. *Structure and Interpretation of Computer Programs*. MIT Press and McGraw-Hill.
- [2] Stephen H. Bach, Victor Sanh, Zheng Xin Yong, Albert Webson, Colin Raffel, Nihal V. Nayak, Abheesh Sharma, Taewoon Kim, M. Saiful Bari, Thibault Férvy, Zaid Alyafeai, Manan Dey, Andrea Santilli, Zhiqing Sun, Srulik Ben-David, Canwen Xu, Gunjan Chhablani, Han Wang, Jason Alan Fries, Maged Saeed AlShaibani, Shanya Sharma, Urmish Thakker, Khalid Almubarak, Xiangru Tang, Dragomir R. Radev, Mike Tian-Jian Jiang, and Alexander M. Rush. 2022. PromptSource: An Integrated Development Environment and Repository for Natural Language Prompts. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics, ACL 2022 - System Demonstrations, Dublin, Ireland, May 22-27, 2022*, Valerio Basile, Zornitsa Kozareva, and Sanja Stajner (Eds.). Association for Computational Linguistics, 93–104. <https://doi.org/10.18653/V1/2022.ACL-DEMO.9>
- [3] Clark W. Barrett, Roberto Sebastiani, Sanjit A. Seshia, and Cesare Tinelli. 2021. Satisfiability Modulo Theories. In *Handbook of Satisfiability - Second Edition*, Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh (Eds.). Frontiers in Artificial Intelligence and Applications, Vol. 336. IOS Press, 1267–1329. <https://doi.org/10.3233/FAIA201017>
- [4] F. L. Bauer and H. Wössner. 1972. The “Plankalkül” of Konrad Zuse: A Forerunner of Today’s Programming Languages. *Comm. ACM* 7 (July 1972).
- [5] Jeremy Bentham. 1791. *The Works of Jeremy Bentham*.
- [6] Luca Beurer-Kellner, Marc Fischer, and Martin T. Vechev. 2023. Prompting Is Programming: A Query Language for Large Language Models. *PLDI* 7 (2023), 1946–1969. <https://doi.org/10.1145/3591300>
- [7] Alan F. Blackwell. 2024. *Moral Codes*. MIT Press.
- [8] Ivan Brakto. 1986. *Prolog Programming for Artificial Intelligence*.
- [9] Gwern Branwen. 2023. GPT-3 Creative Fiction. <https://gwern.net/gpt-3>.
- [10] Saikat Chakraborty and Baishakhi Ray. 2021. On Multi-Modal Learning of Editing Source Code. In *Automated Software Engineering (ASE)*. IEEE, 443–455. <https://doi.org/10.1109/ASE51524.2021.9678559>
- [11] Adam Chlipala. 2013. *Certified Programming with Dependent Types: A Pragmatic Introduction to the Coq Proof Assistant*. MIT Press.
- [12] Garrett Cunningham, Razvan C. Bunescu, and David Juedes. 2022. Towards Autoformalization of Mathematics and Code Correctness: Experiments with Elementary Proofs. (2022), 25–32. <https://aclanthology.org/2022.mathnlp-1.4.pdf>
- [13] Hai Dang, Lukas Mecke, Florian Lehmann, Sven Goller, and Daniel Buschek. 2022. How to Prompt? Opportunities and Challenges of Zero- and Few-Shot Learning for Human-AI Interaction in Creative Applications of Generative Models. *CoRR* abs/2209.01390. <https://doi.org/10.48550/arxiv.2209.01390>
- [14] Leonardo Mendonça de Moura and Nikolaj S. Bjørner. 2008. Z3: An Efficient SMT Solver. In *TACAS*, 337–340. [https://doi.org/10.1007/978-3-540-78800-3\\_24](https://doi.org/10.1007/978-3-540-78800-3_24)
- [15] Edsger W. Dijkstra. 1978. On the foolishness of “natural language programming”. (1978). circulated privately.
- [16] Benj Edwards. 2024. Research AI model unexpectedly attempts to modify its own code to extend runtime. <https://arstechnica.com/information-technology/2024/08/research-ai-model-unexpectedly-modified-its-own-code-to-extend-runtime/>. *Ars Technica* (Aug. 2024).
- [17] Emily First, Markus N. Rabe, Talia Ringer, and Yuriy Brun. 2023. Baldur: Whole-Proof Generation and Repair with Large Language Models. In *Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*, Satish Chandra, Kelly Blincoe, and Paolo Tonella (Eds.). 1229–1241. <https://doi.org/10.1145/3611643.3616243>
- [18] Michel Foucault. 1977. *Discipline and Punish*. Pantheon.
- [19] Simon Frieder, Luca Pinchetti, Ryan-Rhys Griffiths, Tommaso Salvatori, Thomas Lukasiewicz, Philipp Christian Petersen, Alexis Chevalier, and Julius Berner. 2023. Mathematical Capabilities of ChatGPT. *CoRR* abs/2301.13867 (2023). <https://doi.org/10.48550/arXiv.2301.13867>
- [20] Sumit Gulwani, Oleksandr Polozov, and Rishabh Singh. 2017. Program Synthesis. *Founds. Trends. Prog. Lang.* 4, 1-2 (2017), 1–119. <https://doi.org/10.1561/2500000010>
- [21] Michael Townsen Hicks, James Humphries, and Joe Slater. 2024. ChatGPT is bullshit. *Ethics and Information Technology* 26, 38 (2024). <https://doi.org/10.1007/s10676-024-09775-5>
- [22] C.A.R. Hoare. 1981. The Emperor’s Old Clothes. *Commun. ACM* 24, 2 (Feb. 1981).
- [23] Gerard J. Holzmann. 2003. *The SPIN Model Checker: Primer and Reference Manual*. Addison-Wesley.



- [24] Daniel Jackson. 2006. *Software Abstractions: Logic, Language, and Analysis*. MIT Press.
- [25] Cameron Jones and Benjamin Bergen. 2023. Does GPT-4 Pass the Turing Test? [arXiv:2310.20216](https://arxiv.org/abs/2310.20216) [cs.AI]
- [26] Leslie Lamport. 2002. *Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers*. Pearson.
- [27] P. David Lebling, Marc S. Blank, and Timothy A. Anderson. 1979. Zork: A Computerized Fantasy Simulation Game. *IEEE Computer* 12, 4 (April 1979).
- [28] K. Rustan M. Leino. 2023. *Program Proofs*. MIT Press.
- [29] Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. 2023. Making Language Models Better Reasoners with Step-Aware Verifier. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, Toronto, Canada, 5315–5333. <https://doi.org/10.18653/v1/2023.acl-long.291>
- [30] Paqui Lucio. 2016. A Tutorial on Using Dafny to Construct Verified Software. In *XVI Jornadas sobre Programación y Lenguajes PROLE*.
- [31] Jimmy Maher. 2013. Hitchhiking the Galaxy Infocom-Style. [www.filfre.net/2013/11/hitchhiking-the-galaxy-infocom-style/](http://www.filfre.net/2013/11/hitchhiking-the-galaxy-infocom-style/).
- [32] Andrew McChesney. 2014. Great Disappointment Remembered 170 Years On. *Adventist Review* (2014).
- [33] Jordan Meadows and André Freitas. 2022. A Survey in Mathematical Language Processing. *CoRR* (2022). <https://doi.org/10.48550/arXiv.2205.15231>
- [34] Dan Milmo and agencies. 2025. Zuckerberg approved Meta's use of 'pirated' books to train AI models, authors claim. *The Guardian* 10 Jan (2025). <https://www.theguardian.com/technology/2025/jan/10/mark-zuckerberg-meta-books-ai-models-sarah-silverman>.
- [35] Noor Nashid, Mifta Sintaha, and Ali Mesbah. 2023. Retrieval-Based Prompt Selection for Code-Related Few-Shot Learning. In *45th IEEE/ACM International Conference on Software Engineering, ICSE 2023, Melbourne, Australia, May 14-20, 2023*. IEEE, 2450–2462. <https://doi.org/10.1109/ICSE48619.2023.00205>
- [36] Peter Naur and Brian Randell (Eds.). 1968. *Proceedings, NATO Conference on Software Engineering*. Garmisch, Germany.
- [37] George C. Necula and Peter Lee. 1997. Proof-Carrying Code. In *POPL*.
- [38] James Noble. 2024. Automatic Programming vs. Artificial Intelligence. In *AIware*.
- [39] James Noble and Robert Biddle. 2004. Notes on notes on postmodern programming. *ACM SIGPLAN Notices* 39, 12 (2004), 40–56.
- [40] James Noble and Robert Biddle. 2006. Postmodern prospects for conceptual modelling. In *Asia-Pacific Conference on Conceptual Modelling (APCCM)*.
- [41] James Noble and Robert Biddle. 2021. programmingLanguage as Language;.
- [42] Peter Norvig. 1992. *Paradigms of AI Programming: Case Studies in Common Lisp*. Morgan Kaufmann.
- [43] George Orwell. 1948. *Nineteen Eighty-Four*. Secker and Warburg.
- [44] Christine Paulin-Mohring. 2011. Introduction to the Coq Proof-Assistant for Practical Software Verification. In *LASER International Summer School*. Springer.
- [45] Shuofei Qiao, Yixin Ou, Ningyu Zhang, Xiang Chen, Yunzhi Yao, Shumin Deng, Chuanqi Tan, Fei Huang, and Huajun Chen. 2023. Reasoning with Language Model Prompting: A Survey. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, 5368–5393. <https://doi.org/10.18653/v1/2023.ACL-LONG.294>
- [46] Byron Reeves and Clifford Nass. 1996. *The Media Equation: How People Treat Computers, Television, and New Media like Real People and Places*. Cambridge.
- [47] Laria Reynolds and Kyle McDonell. 2021. Prompt Programming for Large Language Models: Beyond the Few-Shot Paradigm. In *CHI '21: CHI Conference on Human Factors in Computing Systems, Virtual Event / Yokohama Japan, May 8-13, 2021, Extended Abstracts*, Yoshifumi Kitamura, Aaron Quigley, Katherine Isbister, and Takeo Igarashi (Eds.). ACM, 314:1–314:7. <https://doi.org/10.1145/3411763.3451760>
- [48] Talia Ringer, Karl Palmskog, Ilya Sergey, Milos Gligoric, and Zachary Tatlock. 2019. QED at Large: A Survey of Engineering of Formally Verified Software. *Found. Trends. Prog. Lang.* 5, 2-3 (2019), 102–281. <https://doi.org/10.1561/25000000045>
- [49] Jean E. Sammet. 1978. The early history of COBOL. In *History of Programming Languages (HOPL)*. Academic Press / ACM, 199–243.
- [50] Marek J. Sergot, Fariba Sadri, Robert A. Kowalski, F. Kriwaczek, Peter Hammond, and H. T. Cory. 1986. The British Nationality Act as a Logic Program. *Commun. ACM* 29, 5 (1986), 370–386. <https://doi.org/10.1145/5689.5920>
- [51] Mary Wollstonecraft Shelley. 1818. *Frankenstein; or the Modern Prometheus*. Lackington, Hughes, Harding, Mavor & Jones.
- [52] James Somers. 2023. Begin : End: A Coder Considers the Waning Days of the Craft. *The New Yorker* (Nov. 2023).
- [53] Michele Tufano, Cody Watson, Gabriele Bavota, Massimiliano Di Penta, Martin White, and Denys Poshyvanyk. 2019. An Empirical Study on Learning Bug-Fixing Patches in the Wild via Neural Machine Translation. *TOSEM* 28, 4 (2019), 19:1–19:29.
- [54] Joseph Weizenbaum. 1966. ELIZA—a computer program for the study of natural language communication between man and machine. *Commun. ACM* 9, 1 (Jan. 1966), 36–45. <https://doi.org/10.1145/365153.365168>
- [55] Joseph Weizenbaum. 1967. Contextual understanding by computers. *Commun. ACM* 10, 8 (Aug. 1967), 474–480. <https://doi.org/10.1145/363534.363545>
- [56] Matt Welsh. 2023. The End of Programming. *Commun. ACM* 66, 1 (Jan. 2023). <https://doi.org/10.1145/3570220>
- [57] Patrick Henry Winston. 1984. *Artificial Intelligence* (2ed ed.). Addison-Wesley.
- [58] Z3Prover. 2023. The Z3 Theorem Prover. <https://github.com/Z3Prover/z3>. [Online]. [Accessed: 2023-09-20].
- [59] Wanqing Zhang. 2024. Dan's the man: Why Chinese women are looking to ChatGPT for love. [www.bbc.com/articles/c4nnje9rpjgo](http://www.bbc.com/articles/c4nnje9rpjgo).