ISSN 2658-5014 (Print) СИИТ ISSN 2686-7044 (Online)
http://siit.ugatu.su 2026. Vol. 8, No. 3(27)
SYSTEMS ENGINEERING AND INFORMATION TECHNOLOGIES
Scientific paper

UDC 004.8 DOI: 10.54708/SIIT-2026-no3-p15 EDN: MJTVAC

Human-Machine Collaborative Deductive Program Synthesis

M. M. Abbasi • M. Joudakizadeh • A. P. Beltiukov

Udmurt State University

Traditional approaches used for deductive program synthesis provide high accuracy of problem analysis and solution generation. However, in order to comprehend and interpret solutions, they require extensive knowledge and thorough understanding of formal logic. This creates a significant barrier between human logic and its implementation in synthesis programs using the deductive approach. This paper proposes a novel human-machine collaborative approach to deductive program synthesis that leverages the mutual strengths of human intuition and advanced logical computation. Our methodology integrates human problem formulation, its automated formalization via large language models, and human verification of the problem, machine-driven program synthesis based on constructive inference, their execution, and iterative feedbacks during the program synthesis. We demonstrate the practical applicability of this approach through an illustrative case study: that is inferring product-manufacturing technologies from a database that is producing and processing the information at the same time. This framework holds the potential for humans to democratize access to powerful formal methods and accelerate the development of complex, logically sound software solutions.

Human–Machine Collaboration; Deductive Program Synthesis; Large Language Models; Natural language processing; Logic Programming; Industrial Automation; Manufacturing Technology Inference.

Introduction

The ambition to automate complex problem-solving mechanisms, related to logical inference, has long been a driving force in the evolution of computer science and artificial intelligence methods. They have seen significant advancements, notably with the rise of large language models (LLMs) employing attention-aware architectures and the enduring power of logic programming. LLMs have demonstrated remarkable capabilities in processing, understanding, and translating natural language, offering a promising avenue for bridging the conceptual gap between human thought and formal specification [Che21]. Concurrently, logic programming, epitomized by languages like Prolog [Col96] and [Bra01] provides robust declarative tools for automated deductive inference.

Despite these strides, a substantial chasm persists between a human's intuitive grasp of a problem and its precise, rigorous formalization required for machine processing. This difference is particularly evident in constructive inference problems, such as program synthesis, where the goal is not merely to verify a statement but to actually construct a program that satisfies it. Existing interactive theorem proving (ITP) systems—like Isabelle/HOL, Lean [De15], and Coq—offer unparalleled accuracy and verifiability. However, their practical applications typically involved specialized knowledge of formal logic and considerable manual effort, limiting their accessibility to a niche community of experts [De94]. This creates a significant barrier for domain specialists who understand the problem intimately but lack the specialized training in formal methods.

Аббаси М. М., Джудакизаде М., Бельтюков А. П. Совместный дедуктивный программный синтез человека и машины // СИИТ. 2026. Т. 8, № 3(27). С. 15-22. (In English). DOI: 10.54708/SIIT-2026-no3-p15. EDN: MJTVAC.

Abbasi M. M., Joudakizadeh M., Beltiukov A. P. “Human-machine collaborative deductive program synthesis” // SIIT. 2026. Vol. 8, no. 3(27), pp. 15-22. DOI: 10.54708/SIIT-2026-no3-p15. EDN: MJTVAC.

This paper introduces a novel paradigm: a human-machine collaborative approach to deductive program synthesis. Our core premise is to harness the complementary strengths of human intuition and advanced logical computation to overcome the limitations inherent in purely manual or fully automated, black-box solutions. We envision a symbiotic pipeline where humans provide the high-level problem understanding and domain knowledge, while machines manage the intricate formalization and rigorous logical deduction. Crucially, this collaboration incorporates explicit human oversight and verification at key stages, mitigating the risks of misinterpretation by automated systems and the complexity overhead of purely manual formalization.

Our methodology integrates several critical steps: human problem articulation, automated formalization of logic powered by natural language processing and transformer models [Her20], human verification of the formal specification, machine-driven program synthesis based on constructive logical inference by [Man92] and [Itz21], execution of the synthesized program, and an iterative feedback loop for refinement. This framework not only aims to democratize access to powerful formal methods but also to accelerate the development of complex, logically sound software solutions by leveraging the emerging capabilities of generative AI in tasks such as theorem proving and code generation according to [Pol20] and [Wan20].

Related Work

Our proposed human-machine collaborative approach to deductive program synthesis builds upon a rich history of research in artificial intelligence, logic programming, formal methods, and recent advancements in large language models (LLMs). In this section, the main focus is to relate our work with the authentic researches already published on the topic of human -machine collaboration for program synthesis. It also highlights the gaps that exits within the published works and describes our approach to cover them.

Deductive Synthesis of Program

It is a fundamental aspect of the human-machine collaboration in which programs are synthesis deductively from their logical specification by using formal proofs. Over the past decade many researchers have published their works on the topic of deductive program synthesis. Among them, Manna and Waldinger were the first one that proposed a framework used for the generation of program from input-output specifications [Man92]. After them, many other researcher followed the same direction of research for deductive program generation. This trend has changed during recent years and the focus of research is shifted towards more complex processes such as the program synthesis by pointers [Itz21] and binary level logical framework for generating the program [Jou24a], [Bel19] and [Abb19]. These approaches requires detailed formal specifications that can be challenge to formulate by a non-expert. To overcome it, over methodology integrate the human intuitions with the automatic formalization of the process of program generation.

Logic Programming and Theorem Proving

Logic programming, exemplified by Prolog, provides a robust paradigm for declarative programming and automated deduction [Bra01]. Its ability to perform constructive inference makes it ideal for program synthesis tasks, systematically searching for solutions that satisfy logical constraints. Recent work has furthered this paradigm, with Joudakizadeh and Bel'tyukov exploring two-level logical formula realization for deductive synthesis, enhancing scalability for complex problems [Jou24a]. Parallel to logic programming, interactive theorem proving (ITP) systems like Lean [De15], Coq, and Isabelle/HOL offer rigorous frameworks for constructing and verifying proofs. Foundational efforts, such as de Bruijn's AUTOMATH, laid the groundwork for verifiable computational proofs [De94]. However, these systems demand significant expertise, a barrier our human-machine collaboration seeks to overcome by automating formalization and incorporating human verification.

Automated theorem proving (ATP) has also seen advancements, with systems such as those outlined by Joudakizadeh and Beltiukov incorporating human feedback into adaptive theorem proving frameworks, which closely resemble our collaborative approach [Jou24b]. Additionally, Hozzová

et al. investigated saturation-based proving for program synthesis, offering a complementary deductive approach [Cod24]. These works reinforce the potential of combining human oversight with automated reasoning, a core principle of our methodology.

Role of Large Language Models in Formalization and Synthesis

The rise of large language models (LLMs) with attention-aware architectures has transformed natural language processing and program synthesis. LLMs trained on code and formal texts excel in tasks like program comprehending and its generation [Che21]. Jain et al.'s Jigsaw framework integrates LLMs with program synthesis to generate programs from natural language, demonstrating the potential for automated formalization [Jai22]. Similarly, Liventsev et al. explored fully autonomous programming with LLMs, reducing human intervention but lacking the iterative verification central to our approach [Liv23]. Feng et al. combined LLMs with enumerative synthesis, improving efficiency in generating correct programs, which informs our use of LLMs for formalizing problem statements [Li24].

Recent studies have furthered LLM applications in synthesis and verification. Mündler et al. developed type-constrained code generation with LLMs, enhancing the precision of synthesized programs, which aligns with our goal of producing logically sound solutions [Mun25]. Wei et al.'s CodeARC benchmark evaluates LLM reasoning for inductive program synthesis, offering insights into human-machine interaction, though our focus remains on deductive methods [Wei25]. Jiang et al.'s SIMCOPILOT evaluates LLMs in copilot-style code generation, highlighting interactive synthesis capabilities relevant to our feedback loop [Jia25]. Additionally, Yang and Sergey explored inductive synthesis of heap predicates, providing a comparative perspective to our deductive approach [Yan25], whereas Zenkner et al.'s transductive synthesis integrates prior knowledge, complementing our iterative refinement process [Zen25].

LLMs also demonstrate potential in formal reasoning and theorem proving. Polu and Sutskever investigated generative language modeling for automated theorem proving, suggesting LLMs can assist in suggesting proof steps [Pol20]. Wang and Deng furthered this by training LLMs to generate theorems, enhancing reasoning capabilities [Wan20]. These advancements support our use of LLMs to bridge the semantic gap between natural language and formal specifications. Techniques like context-free memorization for faster parsing, as proposed by Herman, further optimize the processing of language-based inputs in formalization [Her20].

Other Relevant Domains

Although our work focuses on program synthesis, allied topics provide valuable insights into the collaboration between humans and machines. Coda-Forno et al.'s CogBench explores LLMs in cognitive tasks, providing a framework for evaluating human-like reasoning that could inform our verification step [Cod24]. In robotics, Gökbakan et al.'s data-driven contact estimation for wheeled-biped robots illustrates how machine learning can complement human-defined constraints, analogous to our integration of LLMs and logic [Gök24]. Although not directly related, Singh et al.'s study on modeling circumgalactic media, demonstrates the application of complex simulations to real-world problems, paralleling our case study in manufacturing [Sin24]. Finally, Kupferman and Leshkowitz's work on synthesis with guided environments explores interactive synthesis, offering a perspective on incorporating environmental feedback, similar to our iterative loop [Kup25].

Our methodology uniquely combines human problem articulation, LLM-driven formalization, and deductive synthesis with iterative feedback, addressing the accessibility gap in formal methods while leveraging recent advancements in LLMs and logic programming.

PROPOSED HUMAN-MACHINE METHODOLOGY

Our methodology for deductive program synthesis is a structured, iterative feedback loop that emphasizes collaboration and verification at each critical juncture.

The focus of the proposed methodology is to overcome the limitations of a fully manual approach as well as of a fully automated approach. We accomplish our goal by distributing the tasks of deductive program synthesis between the human and machine. The proposed methodology includes the following steps.

Human Problem Statement

The proposed model is initialized with a human expert defining the problem in the natural language. Its very important how the expert define a complex, ill-formed, real world scenario in a natural language. Its importance lies in the fact that comprehensively defines the inputs and outputs of the problem along with the relationships that exists within the problem. At this stage, the human does not need to concern themselves with formal syntax or logical constructs; the emphasis is purely on conceptual clarity and completeness from a domain perspective. This informal yet rich description serves as the foundational input for the subsequent automated steps.

Machine Formalization

Following the human's natural language description, an Artificial Intelligence (AI) module undertakes the automatic formalization of the problem. This module utilizes advanced natural language processing (NLP) techniques and transformer architectures to interpret the informal problem statement. Its core function is to translate this natural language into a rigorous mathematical model or a precise logical specification. This formalization is typically expressed in a language suitable for constructive predicate logic, ensuring it is unambiguous, machine-readable, and amenable to automated reasoning. This step is crucial for bridging the semantic gap between human language and the strict requirements of logical systems. Let us consider a case study that involves inferring product manufacturing technologies from a database of production and processing information.

During the machine formalization, the formal specifications of the production and processing processes serves as input to the transformation layer of the AI module, which in return generates Prolog clauses.

As follows:

- $P(i) \rightarrow production_{technology}(i, T)$. (1)
- $R(i, j) \rightarrow processing_{technology}(j, i, \text{ProcT})$. (2)
- Implications like $C_n \rightarrow$ Deductive rules with combine_tech. (3)

Human Verification

A critical and distinctive step in our collaborative framework is human verification of the machine-generated formal statement. The human expert reviews the formalized problem to ensure its correctness, fidelity to their original intent, and logical consistency. This active oversight is vital for identifying and correcting any misinterpretations, ambiguities, or errors introduced during the automatic formalization process. By closing the loop between human intuition and machine interpretation at this early stage, we significantly enhance the reliability of the subsequent synthesis. Adjustments and refinements are made as needed, ensuring the formal specification accurately represents the real-world problem.

Machine Program Synthesis (Logical Proof)

Once the formal problem statement has been rigorously verified by a human, a logical machine, preferably based on a declarative logic programming paradigm like Prolog, takes over. The use of Prolog for logical programming in our methodology is due to its ability to handle uncertain or incomplete information. Prolog not only consists of a set of rules and facts that are known to be true, but it can also specify rules and facts that may or may not be true.

To bridge the formal specification to an executable program, the machine program synthesis acts as an intermediate transformation layer. This layer converts the verified logical predicates (e.g., $\mathrm{P}(i), R(i, j)$ and rules like C_n and L_m) into Prolog clauses. For instance:

- Predicates like $P(i)$ are mapped to Prolog facts or rules (e.g., $production_{technology}(I, T)$). (3)

- Implications (e.g., C_n: $(i, j: I, R(i,j), P(j) \Rightarrow P(i))$ become Prolog rules (e.g., $production_{technology}(I,T)$:- $processing_{technology}(J,I,Proc)$, $production_{technology}(J,BaseT)$, combine_tech $(BaseT, Proc, T)$. (4)
- The L_m module for loop prevention is generalized using dynamic cycle detection, such as tabling or path-tracking in Prolog, to handle arbitrary nesting depths without fixed limits.

During program synthesis, the logical machine performs constructive inference to derive a proof or a solution to the problem. Operating strictly under the rules of logic defined by the formal specification, the machine systematically constructs a program that satisfies the stated properties. This step embodies the core of deductive reasoning in program synthesis; the logical machine searches for a sequence of logical derivations that culminate in the desired program, effectively proving its existence and simultaneously generating its structure. The resulting Prolog program is the synthesized executable, which, when queried, performs the logical search to construct results (e.g., manufacturing technologies) deductively.

Program Execution

Upon successful synthesis, the logical specification (or the derived program) is transformed into an executable program. The executable program can be directly executed into in a logical programming environment or can firstly compiled into a more traditional programming language. All this depends upon the target platform and our performance requirements.

The next is to execute the program to achieve our primary goal that is to generate outputs according to the synthesized logic. It practically verifies that the derived program fulfills the intended purpose and produce output as anticipated by the expert.

Iterative Feedback Process

To achieve the maximum accuracy, the entire approach works in an iterative feedback loop. This means in scenarios such as occurrence of an error, results are poor, outputs are undesired, the processes does not terminates.

Instead, it returns to an earlier stage, specifically the problem formulation or formalization revision steps. This feedback loop allows for continuous adjustment and improvement of both the initial problem statement and its formalization. This iterative refinement ensures that the synthesized program accurately addresses the real-world challenges, adapting to new insights or correcting initial misrepresentations. It is a key mechanism for robustness and adaptability in our human-machine collaboration.

CONCLUSION

This paper has presented a human-machine collaborative approach to deductive program synthesis, designed to bridge the persistent gap between human intuitive problem understanding and the rigorous demands of formal logical representation. We have highlighted how traditional deductive synthesis methods, while offering unparalleled accuracy, often necessitate deep expertise in formal logic, thereby creating a significant barrier to their widespread adoption. Our proposed methodology directly addresses this limitation through a carefully orchestrated, iterative feedback loop that strategically leverages the complementary strengths of human cognitive abilities and machine computational precision. This framework integrates several critical stages: human problem formulation in natural language, automated formalization of specifications using advanced large language models, human verification of these machine-generated formalisms, machine-driven program synthesis grounded in constructive logical inference, subsequent execution of the synthesized program, and a vital iterative feedback mechanism for continuous refinement and error correction.

The efficacy and practical utility of this human-machine paradigm were compellingly demonstrated through a detailed case study in the manufacturing industry: inferring product manufacturing technologies from a dynamic database of production and processing information. This application

showcased how the synergistic interaction between human domain knowledge and the deductive power of a logic programming system, specifically exemplified by Prolog, can effectively formalize complex real-world problems. The provided Prolog code illustrated how sophisticated logical rules can be designed to model intricate manufacturing processes, combining basic production technologies with sequential processing steps to constructively derive comprehensive and verifiable product genealogies. This not only yields a solution but also an explicit, logical proof of its derivation, inherent to the deductive nature of the approach.

The distributing of tasks between humans and the machines allows human to work with problems at high level abstraction and domain validations while machine takes care of the tasks such as the precision of formalization of rules and then logical deduction from them. The proposed approach improves the usability and reliability of the formal methodologies. It further moves on providing the real world solutions instead of automated “black box” solutions, which can be at risk of unexpected errors that are difficult to diagnose. The proposed hybrid model is an iterative model that goes through the process of refinement and to make sure that program synthesis during the process accurately reflects the real world dynamics of the problem.

In future, the proposed methodology can be used to handle even more complex natural language tasks. It can be done through advanced LLM techniques along with the improved inference rules that can work on non-sequential processes or the processes that evolve over the time. Similarly, the iterative feedback loop can also be improved to reduce the human efforts in diagnosis of errors during the execution of program. Ultimately, this human-machine cooperative paradigm possesses the tremendous capacity to broaden access to advanced formal verification and synthesis tools, thereby hastening the creation of highly trustworthy and logically rigorous software solutions across a variety of challenging and specialized fields.

REFERENCES

[Abb19] Abbasi M. M., Beltiukov A. Summarizing emotions from text using Plutchik’s Wheel of Emotions // 7th Scient. Conf. Information Technologies for Intelligent Decision Making Support (ITIDS 2019). Atlantis Press, 2019. (Advances in Intelligent Systems ResearchVol. 166). P. 291-294. 10.2991/itids-19.2019.52.

[Bel19] Beltyukov A. P., Abbasi M. M. Logical analysis of emotions in text from natural language // Bulletin of Udmurt University. Mathematics. Mechanics. Computer Science.. 2019. Vol. 29, issue 1. P. 106-116. ZFAVBB.

[Bra01] Bratko I. Prolog programming for artificial intelligence. Pearson education, 2001.

[Che21] Chen M., Tworek J., et al. Evaluating large language models trained on code, 2021. arXiv preprint arXiv:2107.03374. 10.48550/arXiv.2107.03374.

[Cod24] Coda-Forno J., Binz M., et al. CogBench: a large language model walks into a psychology lab. arXiv preprint arXiv:2402.18225. 2024. 10.48550/arXiv.2402.18225.

[Col96] Colmerauer A., Roussel P. The birth of Prolog. History of programming languages—II, 1996. P. 331-367. https://doi.org/10.1145/234286.105782014.

[Cov96] Covington M. A., Nute D., Vellino A. Prolog Programming in Depth. Prentice-Hall, 1996.

[De15] De Moura L., Kong S., et al. The Lean theorem prover (system description). Automated Deduction-CADE-25: 25th Int. Conf. Automated Deduction, Berlin, Germany, August 1-7, 2015, Proc. 25. P. 378-388. 10.1007/978-3-319-21401-6_26.

[De94] De Bruijn, N. G. The mathematical language AUTOMATH, its usage, and some of its extensions // Studies in Logic and the Foundations of Mathematics, 1994. Vol. 133, P. 73-100. 10.1016/S0049-237X(08)70200-3.

[Gök24] Gökbakan Ü. B., Dümbgen F., Caron S. A Data-driven Contact Estimation Method for Wheeled-Biped Robots. 2024. P. 2410.12345. 10.48550/arXiv.2410.12345.

[Her20] Herman G. Faster general parsing through context-free memorization // Proc. 41st ACM SIGPLAN Conf. on Programming Language Design and Implementation, 2020. P. 1022-1035. 10.1145/3385412.3386032.

[Itz21] Itzhaky S., Peleg H., et al. Deductive synthesis of programs with pointers: techniques, challenges, opportunities // Int. Conf. Computer Aided Verification. 2021. P. 110-134. 10.1007/978-3-030-81685-8_5.

[Jai22] Jain N., Vaidyanath S., et al. Jigsaw: Large language models meet program synthesis // Proc. 44th Int. Conf. Software Engineering. 2022. P. 1219-1231. 10.1145/3510003.3510203.

[Jia25] Jiang M., Jain A., et al. SIMCOPILOT: Evaluating Large Language Models for Copilot-Style Code Generation. 2025. 10.48550/arXiv.2505.21514.

[Jou24a] Joudakizadeh M., Bel’tyukov A. P. Two-level realization of logical formulas for deductive program synthesis // Vestn. Udmurtsk. Univ. Mat. Mekh. Komp. Nauki, 2024. Vol. 34, No. 4. P. 469–485. 10.35634/vm240401. JLCQGU.

[Jou24b] Joudakizadeh M., Bel'tyukov A. P. Adaptive human–machine theorem proving system // Izv. IMI UdGU, 64. 2024. P. 17-33. 10.35634/2226-3594-2024-64-02. BVQGKJ.

[Kup25] Kupferman O., Leshkowitz O. Synthesis with guided environments // Int. Conf. Tools and Algorithms for the Construction and Analysis of Systems. 2025. P. 198-216. 10.1007/978-3-031-90653-4_10.

[Li24] Li Y., Parsert J., Polgreen E. Guiding enumerative program synthesis with large language models // Int. Conf. Computer Aided Verification. 2024. P. 280-301. 10.1007/978-3-031-65630-9_15.

[Liv23] Liventsev V., Grishina, A., Härmä A., Moonen L. Fully autonomous programming with large language models. Proceedings of the Genetic and Evolutionary Computation Conference. 2023. P. 1146-1155). 10.1145/3583131.3590481.

[Man92] Manna Z., Waldinger R. Fundamentals of Deduct Program Synthesis (No. SRITN503), 1992.

[Mun25] Mündler N., He J., et al. Type-constrained code generation with language models // Proc. ACM on Programming Languages, 9 (PLDI). 2025. P. 601-626. 10.1145/372927415.

[Pol20] Polu S., Sutskever I. Generative language modeling for automated theorem proving. arXiv preprint arXiv:2009.03393. 2020. 10.48550/arXiv.2009.03393.

[Sin24] Singh P., Lau E. T., et al. Comparison of models for the warm-hot circumgalactic medium around Milky Way-like galaxies // Monthly Notices of the Royal Astron. Society. 2024. Vol. 532. № 3. P. 3222-3235. 10.1093/mnras/stae1695. CYPCAI.

[Wan20] Wang M., Deng J. Learning to prove theorems by learning to generate theorems. Advances in neural information processing systems. 2020. Vol. 33, P. 18146-18157. 10.48550/arXiv.2002.07019.

[Wei25] Wei A., Suresh T., et al. Codearc: Benchmarking reasoning capabilities of LLM agents for inductive program synthesis. 2025. 2503.23145. 10.48550/arXiv.2503.23145.

[Wie12] Wielemaker J., Schrijvers T., et al. Swi-prolog // Theory and Practice of Logic Programming. 2012. Vol. 12. No. 1. P. 67-96. 10.1017/S1471068411000494.

[Yan25] Yang Z., Sergey I. Inductive synthesis of inductive heap predicates // Proc. ACM on Programming Languages, 9(OOPSLA1). 2025. P. 169-195. 10.1145/3720420. IVSWVQ.

[Zen25] Zenkner J., Sesterhenn T., Bartelt C. Transductively Informed Inductive Program Synthesis. 2025. P. 2505. 14744. 10.48550/arXiv.2505.14744.

ОБ АВТОРАХ | ABOUT THE AUTHORS

АББАСИ Мохсин Маншад
Удмуртский государственный университет, Россия.
mohsinmanshad@gmail.com ORCID: 0009-0006-8029-4759.
Канд. техн. наук, доц. кафедры вычислительных технологий и интеллектуальных систем больших данных.

ДЖУДАКИЗАДЕ Милад
Удмуртский государственный университет, Россия.
joudakizadeh@mail.ru ORCID: 0000-0002-6167-6237.
Аспирант, кафедра вычислительных технологий и интеллектуальных систем больших данных.

БЕЛЬТЮКОВ Анатолий Петрович
Удмуртский государственный университет, Россия.
belt.udsu@mail.ru ORCID: 0000-0002-3433-9067.
Д-р физ.-мат. наук, проф., кафедра вычислительных технологий и интеллектуальных систем больших данных.

ABBASI Mohsin Manshad
Udmurt State University, Russia.
mohsinmanshad@gmail.com ORCID: 0009-0006-8029-4759.
Assoc. Prof., Department of Computing Intellectual Systems.

JOUDAKIZADEH Milad
Udmurt State University, Russia.
joudakizadeh@mail.ru ORCID: 0000-0002-6167-6237.
PhD Candidate, Department of Computing Intellectual Systems.

BELTIUKOV Anatoly Petrovich
Udmurt State University, Russia.
belt.udsu@mail.ru ORCID: 0000-0002-3433-9067.
Doctor of Physics and Mathematics, Professor, Department of Computing Intellectual Systems.

МЕТАДАННЫЕ | METADATA

Заглавие: Совместный дедуктивный программный синтез человека и машины.

Авторы: Аббаси М. М., Джудакизаде М., Бельтюков А. П.

Аннотация: Традиционные подходы, используемые для дедуктивного синтеза программ, обеспечивают высокую точность анализа проблем и выработки решений. Однако для понимания и интерпретации решений требуются обширные знания и глубокое понимание формальной логики. Это создает значительный барьер между человеческой логикой и ее реализацией в программах синтеза с использованием дедуктивного подхода. В этой статье предлагается новый совместный подход человека и машины к дедуктивному синтезу программ, который использует взаимные преимущества человеческой интуиции и продвинутых логических вычислений. Наша методология объединяет постановку задачи человеком, ее автоматизированную формализацию с помощью

Title: Human-machine collaborative deductive program synthesis.

Authors: Abbasi M. M., Joudakizadeh M., Beltiukov A. P.

Abstract: Traditional approaches used for deductive program synthesis provide high accuracy of problem analysis and solution generation. However, in order to comprehend and interpret solutions, they require extensive knowledge and thorough understanding of formal logic. This creates a significant barrier between human logic and its implementation in synthesis programs using the deductive approach. This paper proposes a novel human-machine collaborative approach to deductive program synthesis that leverages the mutual strengths of human intuition and advanced logical computation. Our methodology integrates human problem formulation, its automated formalization via large language models, and human verification of the problem,

больших языковых моделей, проверку проблемы человеком, машинный синтез программ на основе конструктивных выводов, их выполнение и итеративные обратные связи в процессе синтеза программы. Мы демонстрируем практическую применимость этого подхода на примере конкретного случая, который заключается в выводе технологий производства продукции из базы данных, которая одновременно генерирует и обрабатывает информацию. Эта структура позволяет людям демократизировать доступ к мощным формальным методам и ускорить разработку сложных, логически обоснованных программных решений.

Ключевые слова: Взаимодействие человека и машины; дедуктивный программный синтез; большие языковые модели; обработка естественного языка; логическое программирование; промышленная автоматизация; вывод производственных технологий.

Язык: Английский.

Статья поступила в редакцию 19.03.2026.

machine-driven program synthesis based on constructive inference, their execution, and iterative feedbacks during the program synthesis. We demonstrate the practical applicability of this approach through an illustrative case study: that is inferring product-manufacturing technologies from a database that is producing and processing the information at the same time. This framework holds the potential for humans to democratize access to powerful formal methods and accelerate the development of complex, logically sound software solutions.

Key words: Human–Machine Collaboration; Deductive Program Synthesis; Large Language Models; Natural language processing; Logic Programming; Industrial Automation; Manufacturing Technology Inference.

Language: English.

The editors received the article on 19 March 2026.