

ON COMBINING ACTIVE KNOWLEDGE CONCEPT AND MACHINE LEARNING

V. A. Perepelkin*, S. B. Arykov**

Institute of computational mathematics and mathematical geophysics SB RAS,

630090, Novosibirsk, Russia

*Novosibirsk State University,

630090, Novosibirsk, Russia

**Novosibirsk State Technical University,

630073, Novosibirsk, Russia

DOI: 10.24412/2073-0667-2026-1-65-78

EDN: YKXJIB

The problem of automatic program construction remains relevant in the context of continuously progressing computerization of all spheres of society. Despite significant achievements in automation tools such as compilers, programming languages, integrated development environments, and artificial intelligence methods, program construction largely remains a creative task that is difficult to automate. Universal approaches to automatic program construction that would produce sufficiently efficient programs for practical use within acceptable timeframes are algorithmically complex and have only niche applications. Therefore, it is important to explore and develop different approaches to automatic program construction and methodologies for their application and combination. In particular, it is valuable to utilize the experience of manual program development in various subject domains and create programming automation tools based on this experience.

This paper examines the active knowledge concept as one such approach and explores how it relates to and can be combined with machine learning methods. The active knowledge concept is a methodology for automatic program construction in specific subject domains. It is based on the theory of synthesis of parallel programs and systems on computational models [1]. The methodology enables automatic construction of sufficiently efficient programs for solving problems of a certain class within a particular subject domain, where efficiency is understood from the perspective of non-functional properties essential to that domain, such as execution time, memory consumption, network load, etc.

The fundamental principles of the active knowledge concept include non-universality, axiomatic theory as a foundation, partial description of the subject domain, use of computational models, multi-level construction, consideration of static and dynamic aspects, application to well-developed subject domains, inclusion of good solutions, variability, recommendations, alternative approach to program creation, accumulation of knowledge in active form, and instrumental support through systems like LuNA (Language for Numerical Algorithms). The LuNA system, developed and maintained at the Institute of Computational Mathematics and Mathematical Geophysics SB RAS, includes a language for describing active knowledge bases and a system for automatic construction and execution of parallel programs.

Machine learning methods, particularly those based on large language models, offer new opportunities for automating programming tasks. A practically widely used technology is the creation

This research was carried out under the state contract with ICM&MG SB RAS FWNM-2025-0005.

of AI assistants — digital assistants capable of not only executing predefined commands but also understanding context and adapting their behavior accordingly. In software development, AI assistants can help with code generation, documentation, quality checking, education, and auxiliary technical tasks. Popular solutions include Cursor, Codex, GitHub Copilot, Gemini Code Assist, and domestic developments like SourceCraft Code Assistant and GigaCode.

This paper explores the potential synergies between the active knowledge concept and machine learning methods. Several principal possibilities for combining these approaches are considered: (1) An AI assistant for training and methodological support of LuNA system users, helping to overcome the entry barrier; (2) AI assistance in creating active knowledge bases, acting as a methodological consultant guiding users through key stages; (3) Automatic creation of active knowledge bases from human-oriented texts using machine learning; (4) AI translation from informal problem statements to formal specifications; (5) Improvement of active knowledge bases through analysis and optimization using large language models.

The paper presents a prototype implementation of an AI assistant for the LuNA system (LuNA AI), focused initially on documentation-related tasks to help overcome the entry barrier in learning and using the active knowledge concept. Technically, LuNA AI is a service that accepts context from the user (files, questions), combines this information with the available knowledge base (documentation, templates, examples), forms a prompt, calls a large language model, and displays the LLM's response to the user.

The implementation utilizes Yandex AI Assistant API, which combines the YandexGPT large language model with RAG (Retrieval-Augmented Generation) technology for retrieving relevant information from knowledge bases. The system architecture includes a frontend application built with Vue and NuxtUI frameworks and a backend developed in Python with FastAPI. The user's query is vectorized and, together with the most relevant context from the knowledge base, is fed to the LLM. For convenient updating of the knowledge base, the concept of a "project" is implemented at the UI level — a set of files forming a knowledge base, with support for multiple projects to maintain different versions of the LuNA system.

The current demonstration version of the AI assistant solves part of the documentation-related tasks: users can request information about the LuNA system, and the AI assistant builds relevant responses using RAG. Although the quality of responses currently requires further improvement, particularly in enhancing the knowledge base quality, even in this form the assistant can provide significant help to those newly acquainted with the LuNA language.

The research demonstrates that the active knowledge concept and machine learning methods have significant potential for combined use. Active knowledge bases, based on axiomatic theories, can serve as a flexible, rigorous mathematical apparatus allowing practically unlimited accumulation of knowledge that remains unchanged over time with the possibility of automatic application. Machine learning methods can rely on this apparatus, solving optimization problems within given axiomatic theories, as well as serving as a bridge to the informal human context.

Future work includes improving the quality of the knowledge base, integrating the AI assistant into popular development environments like VS Code, and expanding its functionality with additional capabilities such as code generation and optimization suggestions within the LuNA framework.

Key words: active knowledge concept, LuNA system, LLM, deep learning, AI assistant.

References

1. Sintez parallelnykh programm i sistem na vychislitelnykh modelyakh / V. A. Valkovsky, V. E. Malyskin; Onv. red. V. E. Kotov; AN SSSR, Sib. otd-nie, VC. — Novosibirsk : Nauka. Sib. otd-nie, 1988. 126 s. (In Russian).

2. Malyshkin V. Active Knowledge, LuNA and Literacy for Oncoming Centuries // LNCS, Vol. 9465. 2015. S. 292-303. DOI: 10.1007/978-3-319-25527-9_19.
3. Malyshkin V. E., Perepelkin V. A. LuNA Fragmented Programming System, Main Functions and Peculiarities of Run-Time Subsystem // Proceedings of the 11th International Conference on Parallel Computing Technologies (PaCT-2011), LNCS, 2011. T. 6873. S. 53–61. DOI: 10.1007/978-3-642-23178-0_5.
4. Arykov S. Defining Order of Execution in Aspect Programming Language // LNCS. 2017. Vol. 10421. P. 255–261. ISBN 978-3-319-62932-2. DOI 10.1007/978-3-319-62932-2_25.
5. Weisz et al. Examining the Use and Impact of an AI Code Assistant on Developer Productivity and Experience in the Enterprise // Proceedings of the Extended Abstracts of the CHI EA '25 Conference on Human Factors in Computing Systems. Article No. 673. P. 1–13. ISBN: 9798400713958. DOI: 10.1145/3706599.3706670.
6. Jiang et al. CursorCore: Assist Programming through Aligning Anything // CoRR. Vol. abs/2410.07002. 2024. ISSN 2331-8422. DOI 10.48550/arXiv.2410.07002.
7. Mark Chen et al. Evaluating Large Language Models Trained on Code // CoRR. Vol. abs/2107.03374. 2021. ISSN 2331-8422. DOI 10.48550/arXiv.2107.03374.
8. Pandey et al. Transforming Software Development: Evaluating the Efficiency and Challenges of GitHub Copilot in Real-World Projects // CoRR. 2024. Vol. abs/2406.17910. ISSN 2331-8422. DOI 10.48550/arXiv.2406.17910.
9. Gemini Code Assist | AI coding assistant [Electron. Res.]: <https://developers.google.com/gemini-code-assist/docs/overview> (accessed 24.11.2025).
10. Grok Code Fast 1 Model Card [Electron. Res.]: <https://data.x.ai/2025-08-26-grok-code-fast-1-model-card.pdf> (accessed 03.02.2025).
11. SourceCraft. Code Assistant. [Electron. Res.]: <https://sourcecraft.dev/portal/code-assistant/> (accessed 22.11.2025).
12. GigaCode [Electron. Res.]: <https://gigacode.ru/> (accessed 22.11.2025).
13. Tal Kadosh et al. OMPar: Automatic Parallelization with AI-Driven Source-to-Source Compilation // CoRR. Vol. abs-2409-14771. 2024. ISSN 2331-8422. DOI 10.48550/arXiv.2409.14771.
14. Mahmud et al. AutoParLLM: GNN-Guided Automatic Code Parallelization using Large Language Models // CoRR. Vol. abs/2310.04047. 2023. ISSN 2331-8422. DOI 10.48550/arXiv.2310.04047.
15. Nichols et al. HPC-Coder: Modeling Parallel Programs using Large Language Models // ISC High Performance 2024 Research Paper Proceedings (39th International Conference). Hamburg, Germany. 2024. P. 1–12. DOI 10.23919/ISC.2024.10528929.
16. Yandex AI Assistant API. [Electron. Res.]: <https://yandex.cloud/ru/docs/ai-studio/concepts/assistant/> (accessed: 24.11.2025).
17. Oche et al. A Systematic Review of Key Retrieval-Augmented Generation (RAG) Systems: Progress, Gaps, and Future Directions // CoRR. 2025. Vol. abs/2507.18910. ISSN 2331-8422. DOI 10.48550/arXiv.2507.18910.

СОВМЕСТНОЕ ПРИМЕНЕНИЕ КОНЦЕПЦИИ АКТИВНЫХ ЗНАНИЙ И МЕТОДОВ МАШИННОГО ОБУЧЕНИЯ

В. А. Перепелкин*, С. Б. Арыков**

Институт вычислительной математики и математической геофизики СО РАН,
630090, Новосибирск, Россия

*Новосибирский национальный исследовательский государственный университет,
630090, Новосибирск, Россия

**Новосибирский государственный технический университет,
630073, Новосибирск, Россия

УДК 004.4'242

DOI: 10.24412/2073-0667-2026-1-65-78

EDN: YKXJIB

В работе рассматриваются возможности совместного использования концепции активных знаний и методов машинного обучения для повышения уровня автоматизации разработки параллельных программ. Концепция активных знаний представляет собой методологию автоматического конструирования программ в конкретных предметных областях. В ее основе лежит идея создания базы активных знаний — специального формализованного описания конкретной предметной области, благодаря чему в этой предметной области существенно снижается сложность автоматического конструирования программ. Методы машинного обучения могут способствовать автоматизации и упрощению создания и использования таких баз активных знаний, снижать порог вхождения в работу с системой автоматического конструирования программ (например, системой LuNA). В работе представлен ИИ-помощник, способный в неформальном общении отвечать на вопросы о системе LuNA и концепции активных знаний. Он реализован на основе большой языковой модели и использует подход RAG для работы с подготовленной для этого текстовой базой знаний. В дальнейшем планируется развивать функциональность ИИ-помощника.

Ключевые слова: концепция активных знаний, система LuNA, машинное обучение, большие языковые модели, ИИ-ассистент.

Введение. На фоне постоянно прогрессирующей компьютеризации практически всех сфер жизни общества не вызывает сомнений актуальность проблемы автоматизации конструирования программ. Она находится под постоянным вниманием ученых и программистов, и ее разработка весьма плодотворна с точки зрения того, какое количество ручного труда удастся переложить на компьютер с приемлемым (а зачастую и превосходящим ручное) качеством этого труда. К таким результатам можно отнести компиляторы, языки и системы программирования, интегрированные среды разработки, генераторы документации, методы искусственного интеллекта и т. п.

Несмотря на колоссальные усилия и результаты в этой области, проблема автоматического конструирования программ на текущий момент остается открытой, и запрос на

Работа выполнена в рамках государственного задания ИВМиМГ СО РАН FWNM-2025-0005.

развитие средств автоматизации программирования только возрастает. Это связано с тем, что в значительной степени конструирование программ остается творческой задачей, плохо поддающейся автоматизации, а прямолинейные и универсальные подходы к автоматическому конструированию за приемлемое время достаточно хороших для практического использования программ являются алгоритмически труднорешаемыми и потому имеют только нишевое практическое применение.

Как следствие, важно исследовать и развивать разные подходы к автоматическому конструированию программ и методологию их применения и сочетания. В частности, важно использовать опыт ручной разработки программ в различных предметных областях и на его основе создавать средства автоматизации программирования. В статье рассматривается один из таких подходов — концепция активных знаний — с точки зрения того, как он соотносится и как может сочетаться с некоторыми методами машинного обучения. Также в статье представлены текущие результаты по реализации одного из таких возможных сочетаний — созданию программного ассистента на основе больших языковых моделей для системы активных знаний LuNA.

Остальная часть статьи структурирована следующим образом. В разделе 1 приводится принципиальное описание концепции активных знаний. В разделе 2 дается характеристика методам искусственного интеллекта, затрагиваемых в статье. В разделе 3 рассматриваются возможности совместного использования концепции активных знаний и методов машинного обучения. В разделе 4 представлены текущие результаты по созданию ИИ-помощника для системы LuNA. В заключении подводятся итоги статьи.

1. О концепции активных знаний. Концепция активных знаний — это методология автоматического конструирования программ в конкретных предметных областях. Она базируется на теории синтеза параллельных программ и систем на вычислительных моделях [1] и была изложена в [2]. Эта методология позволяет обеспечить автоматическое конструирование достаточно эффективных программ решения задач определенного класса в некоторой предметной области. Тут и далее эффективность понимается с точки зрения существенных в той или иной предметной области нефункциональных свойств программы, таких как время выполнения, расход памяти, нагрузка на сеть и т. п. Под функциональными свойствами в работе будем понимать функцию, которую вычисляет программа, т. е. как она отображает свои входные данные в выходные.

Рассмотрим кратко основные принципы концепции активных знаний, дающие представление об этой методологии в целом.

Неуниверсальность. В общей постановке задача автоматического конструирования оптимальных программ по их высокоуровневой спецификации является алгоритмически труднорешаемой задачей, поэтому поиск универсального способа нецелесообразен. Тем не менее, во многих предметных областях существуют различные частные и эвристические подходы и алгоритмы, способные обеспечить за разумное время конструирование достаточно хороших для практического применения программ. Для этого требуется построить базу активных знаний — формализованное частичное описание предметной области, которое содержит информацию о существенных с точки зрения автоматического конструирования программ особенностях этой предметной области и накопленных в ней существующих программных модулях решения отдельных подзадач. Существование достаточно хорошей базы активных знаний позволяет системе конструирования программ (например, системе LuNA [3]) автоматически конструировать в этой предметной обла-

сти программы удовлетворительного качества. Чем уже предметная область, тем проще построить для нее базу активных знаний.

Аксиоматическая теория в основе. В основе базы активных знаний лежит аксиоматическая теория, которая частично описывает предметную область. Аксиоматическая теория представляет собой универсальный математический инструмент, позволяющий строго описывать практически что угодно в предметной области и однозначно интерпретировать это описание, в том числе с помощью компьютера. Наличие такого формального описания позволяет строить описания предметных областей, которые могут быть использованы неограниченно долго в будущем, что позволяет использовать базы активных знаний как способ накопления информации о существующих решениях (алгоритмах, программах, математических моделях, методах и т. п.) для их автоматического использования в решении новых задач.

Частичность описания предметной области. Аксиоматическая теория описывает не всевозможные детали о предметной области, а только частично. В аксиоматическую теорию входит описание ключевых понятий предметной области; хороших, проверенных практикой и отлаженных программных модулей; сложившейся практики применения этих модулей для решения практических задач; существующих эвристик и частных алгоритмов, обеспечивающих конструирование достаточно эффективных программ. Другие детали в аксиоматической теории не описываются, чтобы множество комбинаций не становилось слишком большим, и база активных знаний могла использоваться прагматично.

Использование вычислительных моделей. В качестве представления аксиоматических теорий используются вычислительные модели (см. [1]). Вычислительная модель является математической моделью, позволяющей описывать аксиоматические теории частного вида не с помощью аксиом и правил вывода, а в представлении, подобном двудольному ориентированному графу, заданному параметрически. Вычислительная модель прямо описывает понятия предметной области в виде множества переменных, могущих иметь значения. Также она включает множество операций, описывающих возможность вычислять значения одних понятий из других с помощью вычислительных модулей, называемых фрагментами кода. В качестве фрагментов кода могут выступать готовые программные модули — библиотечные процедуры, программы и т. п., разработанные на традиционных языках программирования.

Использование вычислительных моделей вместо классического представления аксиоматической теории в виде множества аксиом и правил вывода имеет прагматический смысл. Он заключается в том, что даже незначительное изменение аксиоматической теории (добавление, удаление или модификация какого-либо правила вывода или аксиомы) может кардинально и труднопредсказуемо изменить множество ее теорем. Изменение же вычислительной модели локально и гораздо легче понять и оценить последствия такого изменения. Это, как правило, добавление или удаление какого-либо понятия (переменной) или способа вычислять одни понятия из других (операция). Также теория синтеза параллельных программ и систем на вычислительных моделях уже хорошо развита в части автоматического конструирования алгоритмов и программ с учетом их нефункциональных свойств.

Многоуровневое конструирование. Процесс конструирования программы логически (а не хронологически) разделяется на несколько этапов. На первом уровне имеется постановка задачи и база активных знаний. Далее на этой основе строится алгоритм решения задачи (в [1] — планирование, построение VW-плана). На следующем уровне строится

отображение операций и переменных алгоритма на ресурсы параллельного вычислителя и определяется управление, которое задает частичный порядок выполнения операций [4]. Это задает программу решения задачи. Последний уровень — это исполнение программы, на котором обеспечиваются ее динамические свойства исполнительной системой или интерпретатором.

Такое разделение на уровне задает декомпозицию задачи автоматического конструирования программы на подзадачи (вывод алгоритма, генерация программы, обеспечение динамических свойств), сводя решение задачи к решению подзадач, и тем самым снижая сложность процесса автоматического конструирования программ.

Статика и динамика. Процесс конструирования и исполнения программы можно рассматривать как процесс принятия решений. Исходно, когда определена только задача, то на базе активных знаний может быть выведено, вообще говоря, множество различных решений (реализаций). Когда фиксируется алгоритм решения задачи, то множество реализаций сокращается до тех, которые реализуют выбранный алгоритм. И так далее, по мере принятия решений — о распределении ресурсов, управлении и т. п. — множество реализаций сокращается до одноэлементного, соответствующего фактически реализованному.

С этой точки зрения все решения могут приниматься как статически, на этапе конструирования кода программы, так и динамически — во время исполнения. В пределе это дает генерацию полностью статического кода или полностью динамическую интерпретацию. Промежуточный вариант означает исполнение частично определенной программы с помощью исполнительной системы, обеспечивающей динамические свойства (динамическая сборка мусора, динамическая балансировка нагрузки на узлы и т. п.). По возможности все решения следует принимать статически, чтобы минимизировать накладные расходы во время исполнения, а на время исполнения оставлять принятие только тех решений, которые необходимо принимать динамически для обеспечения динамических свойств.

Освоенные предметные области. Концепцию активных знаний следует применять для более или менее освоенных (разработанных) предметных областей — то есть таких, в которых уже имеются накопленные программные модули, которые могут быть переиспользованы в качестве решения подзадач при решении новых задач в этой предметной области, уже сложилась практика применения этих модулей, выработались необходимые эвристики, обеспечивающие хорошую эффективность программ, и т. п. Без этого материала в базу активных знаний будет нечего включать, и ее полезность будет сомнительна. Выгода по сравнению с ручным программированием заключается в том, что эти знания теперь используются в процессе конструирования программы автоматически, без необходимости человека их осваивать. Именно это и понимается под термином «активные знания» в противоположность «пассивным знаниям», примером которых может служить учебник — для применения содержащихся в нем знаний необходим человек, который их освоит.

Включение хороших решений. Так как конструирование новых программ в концепции активных знаний состоит в комбинировании уже имеющихся программных модулей в рамках, определенных аксиоматической теорией, то база активных знаний должна содержать достаточно таких модулей для того, чтобы охватить класс практически значимых задач. Качество результирующей программы существенно зависит от качества модулей, включенных в базу активных знаний. Качество результирующей программы зависит не только от качества модулей, но и от аксиоматической теории, и от алгоритмов конструирования программы.

Вариативность. В базу активных знаний включается множество различных решений одних и тех же задач/подзадач. Решения должны различаться между собой своими нефункциональными свойствами, и эти нефункциональные свойства должны быть формально описаны в базе активных знаний. Благодаря этому система конструирования сможет отдавать предпочтения тем вариантам, которые лучше подходят под ситуацию — особенности и конфигурацию аппаратного обеспечения, свойства обрабатываемых данных, состояние загруженности вычислителя и т. п.

Рекомендации. Программа конструируется автоматически, но возможно вмешательство человека в этот процесс для его настройки. Концепция активных знаний предполагает возможность включения в базу активных знаний рекомендаций — формально описанных высокоуровневых предложений о том, как следует конструировать программу в этой предметной области. Примером может служить формула, отображающая множество операций и переменных алгоритма на узлы мультимпьютера, или заданное отношение частичного порядка на множестве операций, определяющая рекомендуемый частичный порядок их исполнения. Конструирование достаточно эффективной программы — сложная задача, часто требующая творческого подхода к своему решению. Рекомендации позволяют включить в базу активных знаний интуитивные представления человека о том, как следует конструировать и исполнять программу, существенно расширив область применения концепции активных знаний и повысив качество конструируемых программ. В перспективе, когда системные алгоритмы будут превосходить по качеству советы человека, рекомендации могут быть безболезненно отброшены из базы активных знаний.

Подход к созданию программ. Концепция активных знаний предлагает альтернативный традиционному подход к созданию программ. Вместо программирования как такового во главу угла становится математическая логика, которая используется для выражения знаний о предметной области и постановке задачи. Если на существующей базе активных знаний нужно решение построить невозможно, или оно оказывается недостаточно эффективным, то это решается включением в базу активных знаний новых элементов (модулей, алгоритмов, понятий и т. п.), а не дописыванием или переписыванием программы.

Накопление знаний в активной форме. Концепция активных знаний дает основу для практически неограниченного накопления знаний в активной форме с целью их последующего автоматического использования в решении новых задач. Это достигается за счет того, что база активных знаний — это отчуждаемое представление знаний, которое может быть проанализировано людьми (или компьютером): модули и аксиоматические теории, проверенные временем и практикой, включаются в базы активных знаний; в случае обнаружения ошибок — исключаются; более эффективные модули включаются в базы активных знаний, со временем вытесняя менее эффективные альтернативы. Этот процесс подобен научному освоению предметных областей, когда со временем полезные и практические понятия, теоремы и методы накапливаются в арсенале научного сообщества, а ошибочные теории и менее эффективные решения постепенно выявляются, опровергаются, исключаются из обихода. Ключевым тут является отчуждаемость и однозначная интерпретация знаний, представленных в виде базы активных знаний, их неизменность во времени. Это отличает концепцию активных знаний от некоторых методов машинного обучения, где затруднительно выявить и подвергнуть анализу знания, заложенные в искусственную нейросеть, а дообучение нейронной сети не гарантирует неизменность предыдущих знаний.

Система LuNA. Концепция активных знаний как методология может быть поддержана инструментально. Примером такого инструмента является система LuNA [3] (от Language for Numerical Algorithms — язык для численных алгоритмов), созданная и развиваемая в ИВМиМГ СО РАН. Эта система включает в себя одноименный язык, позволяющий описывать базы активных знаний, и систему автоматического конструирования и исполнения параллельных программ.

2. Разработка программ с применением методов машинного обучения. Активно применяемой на практике технологией на основе машинного обучения является создание ИИ-ассистентов (или ИИ-помощников). По сути, это цифровой помощник, умеющий не только выполнять набор заранее заданных команд, но понимающий контекст и меняющий свое поведение с его учетом.

Современные ИИ-ассистенты могут видеть, слышать, читать и работать с цифровыми инструментами. Они находят широкое применение как в личном использовании (например, могут разобрать почту или запланировать путешествие), так и в производственных задачах (например, выполняют функции тех. поддержки первой линии). Применительно к разработке программного обеспечения, ИИ-ассистент — это инструмент для помощи в выполнении задач, связанных с кодированием, документированием и тестированием кода [5]. Далее везде под ИИ-ассистентом будем иметь в виду именно помощника по разработке ПО.

Рассмотрим подробнее, как ИИ-ассистент может быть полезен в процессе разработки:

1) Работа с кодом: создать шаблон модуля или класса, предложить автодополнение или оптимизацию на основе контекста (завершение текущей строки или даже всей функции), дать рекомендации по архитектуре, сделать рефакторинг или просто объяснить код.

2) Работа с документацией: дать разъяснения о функционировании проекта, рекомендовать архитектурные решения на основе принятых в проекте стандартов, сгенерировать описание функции.

3) Проверка качества: выявление подозрительных (с точки зрения наличия ошибок) фрагментов кода, генерация юнит-тестов, выполнение UI-тестов, создание тестовых фикстур и моков.

4) Обучение: объяснение сложных концепций языков программирования или нюансов использования библиотек, подбор примеров, сравнительный анализ альтернатив.

5) Вспомогательные технические задачи: генерация конфигурационных файлов, отправка кода в репозиторий и другие.

Представленный перечень не является исчерпывающим, однако позволяет сделать вывод, что спектр помощи, оказываемой ИИ-ассистентом, весьма широк, и потенциально он может значительно повысить продуктивность разработчика.

Существует целый ряд популярных ИИ-ассистентов для промышленной разработки программного обеспечения. К наиболее известным зарубежным решениям можно отнести Cursor [6], Codex [7], GitHub Copilot [8], Gemini Code Assist [9], Grok Code Fast [10], watsonx [5]. Российские инструменты представлены такими решениями как SourceCraft Code Assistant [11] и GigaCode [12].

Ведутся исследовательские проекты по разработке ИИ-ассистентов для специализированных применений. В частности, в области параллельных вычислений можно отметить OMPar [13], AutoParLLM [14], HPC-Coder [15].

Когда дело доходит до высокой производительности (а в параллельных вычислениях это всегда так), специализированные решения для выбранной предметной области име-

ют преимущество перед универсальными системами, так как при их построении можно учесть специфические знания о предметной области и использовать эти знания для оптимизации. Далее рассматриваются некоторые особенности концепции активных знаний, которые могут лечь в основу специализированного ИИ-ассистента.

3. Концепция активных знаний и методы машинного обучения. Рассмотрим некоторые принципиальные возможности сочетания методов машинного обучения и концепции активных знаний.

ИИ-помощник в обучении и методическом сопровождении пользователей системы LuNA. Знакомство с новым языком и системой конструирования программ связано с преодолением порога вхождения, что включает знакомство с терминологией, освоение ключевых принципов и понятий, знакомство с инструментарием и сложившимися практиками. В этой области хорошо могут помочь ИИ-помощники (ассистенты, агенты), которые могут служить связующим звеном между текущим уровнем понимания пользователя и базовым пониманием системы LuNA. ИИ-помощник может помочь в интерпретации сообщений об ошибках, подсказать следующий шаг, ответить на вопрос пользователя, направить его в нужный раздел документации или предложить ознакомиться с публикацией по теме.

Создание базы активных знаний — это сложная задача, имеющая свою специфику в каждой предметной области. Этот процесс имеет свои ключевые этапы и методы выявления и решения задач — определение понятий и концептуализация предметной области, выделение фрагментов кода из имеющегося ПО, определение существенных нефункциональных свойств и способов их представления в базе активных знаний и т. д. ИИ-помощник может существенно помочь в этом процессе, выступив в качестве методического консультанта, проводя пользователя через эти этапы создания базы активных знаний. Отдаленно это можно сравнить с ИИ-консультантом по распараллеливанию последовательных программ, но в данном случае создается не параллельная программа, а база активных знаний, и создается она из существующего программного обеспечения и знаний экспертов в этой предметной области.

Автоматическое создание баз активных знаний. Методы машинного обучения могут использоваться для автоматического создания базы активных знаний из человеко-ориентированных текстов (например, учебников, статей, описаний алгоритмов и т. п.). В простом случае для этого формируется запрос, включающий фрагмент человеко-ориентированного текста, информации о том, что такое база активных знаний и как она представляется на конкретном языке. Так как концепция активных знаний содержит методологию создания баз активных знаний, то этой методологии при определенных ограничениях может следовать и искусственный интеллект. Это направление работ имеет большой потенциал к автоматизированному переводу знаний из пассивной в активную форму. Также в качестве исходного материала могут использоваться не только человеко-ориентированные тексты, но и существующее программное обеспечение.

ИИ-перевод из неформальной постановки задачи в формальную. Решение задач на основе готовой базы активных знаний требует формальной постановки задачи, которую требуется решить. Методы машинного обучения могут быть использованы для перевода постановки задачи из неформального вида в формальный. То же самое касается и интерпретации результатов (решения задачи) — оно может быть переведено языковой моделью в человеко-ориентированную форму, в том числе в ту терминологию, в которой задача была неформально поставлена. Также неформальная постановка задачи может быть

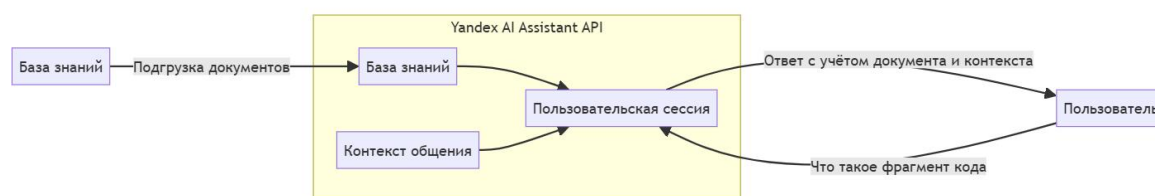


Рис. 1. Архитектура ИИ-ассистента

использована для поиска ИИ-помощником подходящей для этой задачи базы активных знаний.

Улучшение баз активных знаний. Методы машинного обучения, в частности, большие языковые модели, могут использоваться для анализа и улучшения баз активных знаний в том же ключе, как если бы это делал инженер знаний: выявление заведомо избыточных элементов в базе активных знаний, формирование сценариев тестирования работоспособности и эффективности конструируемых программ, определение дублирования и выявление тождественных понятий в различных базах активных знаний.

Также конструирование программ на основе баз активных знаний позволяет строить правильные программы (при условии отсутствия ошибок в самой базе активных знаний), но обеспечение качества конструируемой программы — это сложная оптимизационная проблема, требующая экспертных знаний. Методы машинного обучения могут использоваться для ее решения, имитируя анализ и принятие решений человека при ручной реализации этой же задачи. Построенное таким образом решение может оказаться не самым эффективным или неэффективным вовсе, но, по крайней мере, оно будет работать правильно. Во многих же случаях методы ИИ смогут находить таким образом вполне разумные способы решения задачи.

Подводя итоги разделу, можно сказать, что концепция активных знаний и машинное обучение имеют потенциал к совместному использованию. Базы активных знаний, базируясь на аксиоматических теориях, могут выступать в качестве гибкого строгого математического аппарата, позволяющего практически неограниченно накапливать знания, которые будут сохраняться неизменно во времени с возможностью из автоматического применения. Методы машинного обучения могут опираться на него, решая оптимизационные задачи в рамках заданных аксиоматических теорий и выступая связующим звеном с неформальным контекстом человека.

4. ИИ-ассистент для системы LuNA. Для системы LuNA первостепенной задачей представляется помощь в преодолении «входного барьера» при освоении и использовании концепции активных знаний. Поэтому в первой версии ИИ-ассистента LuNA (далее — LuNA AI) было решено сосредоточиться на задачах, связанных с работой с документацией.

Технически LuNA AI представляет собой сервис, который принимает от пользователя контекст (файлы, вопросы), при необходимости комбинирует эту информацию с имеющейся базой знаний (документацией, шаблонами, примерами), формирует промт, вызывает большую языковую модель (LLM) и отображает пользователю ответ LLM.

Для экономии времени на разработку LuNA AI построен с использованием Yandex AI Assistant API [16], который объединяет большую языковую модель YandexGPT и технологию RAG [17], добавляющую возможности поиска и извлечения релевантной информации из больших баз данных или документов.

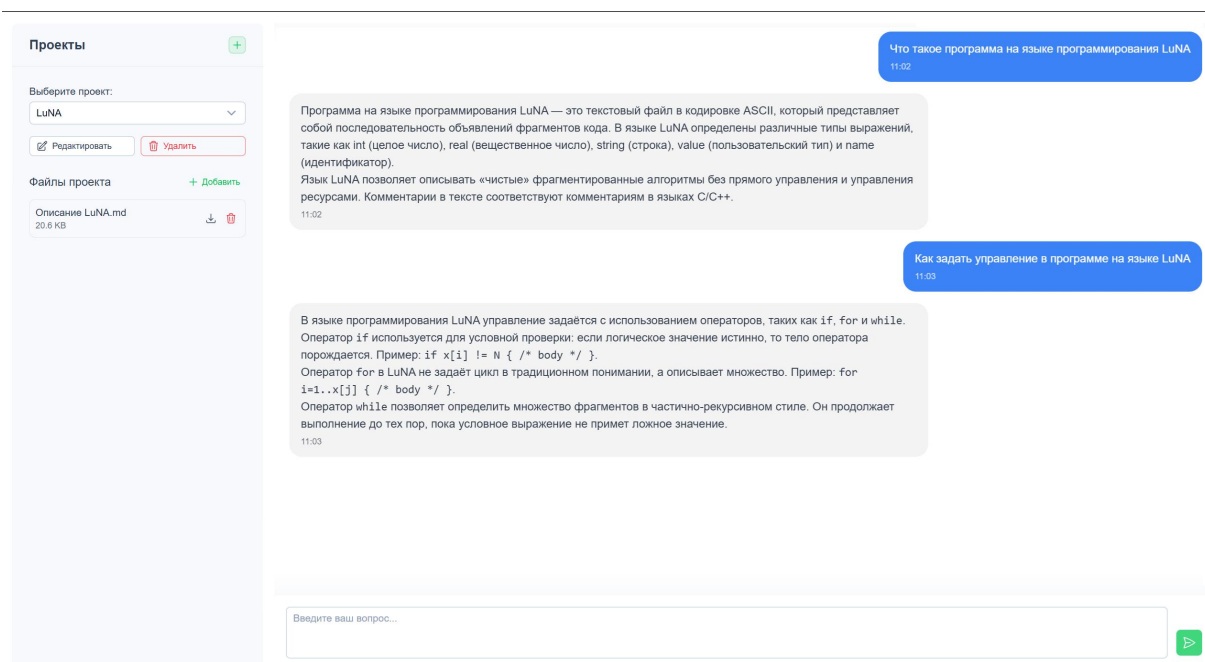


Рис. 2. Пример работы с LuNA AI

Для встраивания ИИ-ассистента в процесс разработки может создаваться отдельная IDE (как в случае с Cursor), ИИ-ассистент может являться плагином (GitHub Copilot или Gemini Code Assist), либо отдельным сервисом, основанным на универсальной (GPT-5.1, Gemini 3 Pro и др.) или специализированной для разработки программного кода (DeepSeek Coder, Qwen3-Coder и др.) модели. В последнем случае требуется дополнительная ручная работа: пользователь самостоятельно формирует запрос к модели, а затем, при необходимости, самостоятельно переносит результат в IDE. С учетом ориентации на задачи по работе с документацией, первая версия LuNA AI реализована в форме отдельного сервиса.

Реализована демонстрационная версия ИИ-ассистента, решающая часть задач по работе с документацией: пользователь может запрашивать информацию о системе LuNA, а ИИ-ассистент с использованием RAG строит релевантный ответ.

Архитектура сервиса представлена на рис. 1. На основе существующей документации по системе LuNA была сформирована база знаний и загружена в созданное в Yandex Cloud пространство через API. Запрос пользователя обрабатывается фронтенд-приложением, построенным с помощью фреймворка Vue и библиотеки NuxtUI, и передается посредством REST-запроса на бэкенд, разработанный на языке Python с применением FastAPI. Далее запрос векторизуется и вместе с наиболее релевантным контекстом из базы знаний подается на вход LLM.

Пример работы с LuNA AI приведен на рис. 2. Пользователь задает вопросы о системе программирования LuNA, а ИИ-ассистент дает ответы.

Для удобного обновления базы знаний на уровне UI реализовано понятие «проекта» — набора файлов, формирующего базу знаний. Проектов может быть несколько. Таким образом можно поддерживать несколько версий системы LuNA, которые значительно отличаются друг от друга.

Заключение. В работе принципиально рассмотрены концепция активных знаний и некоторые ИИ-подходы с точки зрения автоматического конструирования программ. Рас-

смотрено, как они могут сочетаться, дополняя друг друга. Представлена демонстрационная версия ИИ-ассистента, обеспечивающая работу с документацией по системе LuNA.

На текущий момент качество выдаваемых ИИ-ассистентом ответов не является оптимальным и требует дальнейшего улучшения, прежде всего, в части повышения качества базы знаний. Однако даже в таком варианте ИИ-ассистент способен оказывать существенную помощь тем, кто впервые знакомится с языком LuNA.

В настоящий момент ИИ-ассистент реализован как отдельный сервис. В дальнейшем планируется интегрировать его в популярную открытую среду разработки VS Code, а также расширить его функционал дополнительными возможностями.

Список литературы

1. Вальковский В. А., Малышкин В. Э. Синтез параллельных программ и систем на вычислительных моделях / Отв. ред. Котов В. Е.; АН СССР, Сиб. отд-ние, ВЦ. Новосибирск: Наука. Сиб. отд-ние, 1988. 126 с.
2. Malyshkin V. Active Knowledge, LuNA and Literacy for Oncoming Centuries // LNCS, Vol. 9465. 2015. С. 292–303. DOI: 10.1007/978-3-319-25527-9_19.
3. Malyshkin V. E., Perepelkin V. A. LuNA Fragmented Programming System, Main Functions and Peculiarities of Run-Time Subsystem // Proceedings of the 11th International Conference on Parallel Computing Technologies (PaCT-2011), LNCS, T. 6873, 2011, С. 53–61. DOI: 10.1007/978-3-642-23178-0_5.
4. Arykov, S. Defining Order of Execution in Aspect Programming Language // LNCS. 2017. Vol. 10421. P. 255–261. ISBN 978-3-319-62932-2. DOI: 10.1007/978-3-319-62932-2_25.
5. Weisz et al. Examining the Use and Impact of an AI Code Assistant on Developer Productivity and Experience in the Enterprise // Proceedings of the Extended Abstracts of the CHI EA '25 Conference on Human Factors in Computing Systems. Article No. 673. P. 1–13. ISBN: 9798400713958. DOI: 10.1145/3706599.3706670.
6. Jiang et al. CursorCore: Assist Programming through Aligning Anything // CoRR. Vol. abs/2410.07002. 2024. ISSN 2331–8422. DOI: 10.48550/arXiv.2410.07002.
7. Mark Chen et al. Evaluating Large Language Models Trained on Code // CoRR. Vol. abs/2107.03374. 2021. ISSN 2331–8422. DOI: 10.48550/arXiv.2107.03374.
8. Pandey et al. Transforming Software Development: Evaluating the Efficiency and Challenges of GitHub Copilot in Real-World Projects // CoRR. Vol. abs/2406.17910. 2024. ISSN 2331–8422. DOI: 10.48550/arXiv.2406.17910.
9. Gemini Code Assist | AI coding assistant. [Электронный ресурс]: <https://developers.google.com/gemini-code-assist/docs/overview> (дата обращения: 24.11.2025).
10. Grok Code Fast 1 Model Card. [Электронный ресурс]: <https://data.x.ai/2025-08-26-grok-code-fast-1-model-card.pdf> (дата обращения: 03.02.2025).
11. SourceCraft. Code Assistant. [Электронный ресурс]. <https://sourcecraft.dev/portal/code-assistant/> (дата обращения: 22.11.2025).
12. GigaCode. [Электронный ресурс]. <https://gigacode.ru/> (дата обращения: 22.11.2025).
13. Tal Kadosh et al. OMPar: Automatic Parallelization with AI-Driven Source-to-Source Compilation // CoRR. Vol. abs-2409-14771. 2024. ISSN 2331-8422. DOI: 10.48550/arXiv.2409.14771.
14. Mahmud et al. AutoParLLM: GNN-Guided Automatic Code Parallelization using Large Language Models // CoRR. Vol. abs/2310.04047. 2023. ISSN 2331–8422. DOI: 10.48550/arXiv.2310.04047.
15. Nichols et al. HPC-Coder: Modeling Parallel Programs using Large Language Models // ISC High Performance 2024 Research Paper Proceedings (39th International Conference). Hamburg, Germany. 2024. P. 1–12. DOI: 10.23919/ISC.2024.10528929.

16. Yandex AI Assistant API. [Электронный ресурс]: <https://yandex.cloud/ru/docs/ai-studio/concepts/assistant/> (дата обращения: 24.11.2025).

17. Oche et al. A Systematic Review of Key Retrieval-Augmented Generation (RAG) Systems: Progress, Gaps, and Future Directions // CoRR. Vol. abs/2507.18910. 2025. ISSN 2331-8422. DOI: 10.48550/arXiv.2507.18910.



Перепелкин Владислав Александрович — кандидат технических наук, старший научный сотрудник Института вычислительной математики и математической геофизики СО РАН; доцент кафедры параллельных вычислений факультета информационных технологий Новосибирского государственного университета. Тел.: (383) 330-89-94, e-mail: perepelkin@ssd.sccc.ru. В 2008 году окончил Новосибирский государственный университет с присуждением степени магистра техники и технологии по направлению «Информатика и вычислительная техника». В 2023 году защитил кандидатскую диссертацию. Имеет более 20 опубликованных статей по теме автоматизации конструирования параллельных программ в области численного моделирования. Является одним из основных разработчиков экспериментальной системы автоматического конструирования численных параллельных программ для мультикомпьютеров LuNA (от Language for Numerical Algorithms). Область профессиональных интересов включает автоматизацию конструирования параллельных программ, языки и системы параллельного программирования, высокопроизводительные вычисления.

Perepelkin Vladislav Aleksandrovich — PhD in Computer Science. Graduated from Novosibirsk State University in 2008 with the master degree in engineering and technology in computer science. Defended his PhD thesis in 2023. Nowadays has the senior researcher position at the Institute of Computational Mathematics and Mathematical Geophysics of the Siberian Branch of Russian Academy of Sciences, and also has a part time associated professor position in the Novosibirsk State University. He is an author of

more than 20 papers on automation of numerical parallel programs construction. He is one of main developers of system LuNA (Language for Numerical Algorithms) for automatic construction of numerical parallel programs. Professional interests include automation of parallel programs construction, languages and systems of parallel programming, high performance computing.



Арыков Сергей Борисович — кандидат физико-математических наук, младший научный сотрудник Института вычислительной математики и математической геофизики СО РАН; доцент кафедры параллельных вычислительных технологий факультета прикладной математики и информатики Новосибирского государственного технического университета. E-mail: arykov@sscc.ru. Окончил Новосибирский государственный университет. Автор более 20 научных работ. Область профессиональных интересов включает автоматизацию конструирования параллельных программ, языки и системы параллельного программирования, технологии искусственного интеллекта.

Sergey Arykov — PhD in Computer Science, a junior researcher at the Institute of Computational Mathematics and Mathematical Geophysics of the Siberian Branch of the Russian Academy of Sciences. He is also an associated professor at the Department of Parallel Computing Technologies, Faculty of Applied Mathematics and Computer Science, Novosibirsk State Technical University. E-mail: arykov@sscc.ru. Graduated from Novosibirsk State University. The author of more than 20 scientific publications. Professional interests include the automation of parallel program design, parallel programming languages and systems and artificial intelligence technologies.

Дата поступления — 04.02.2026