Online Methods for An AI agent for treatment reasoning over a biomedical tool universe

Shanghua Gao[1], Ayush Noori[1,2,3], Richard Zhu[1], Curtis Ginder[1,4], Zhenglun Kong[1], Xiaorui Su[1], Justin Kauffman[5], Benjamin S. Glicksberg[5,6,7], Joshua Lampert[5,6,8], Ankit Sakhuja[5,9,10], Ashwin Sawant[5,9,11], ATHENA-R1 Evaluation Consortium[12], David A. Clifton[2,13], Noa Dagan[3,14,15], Ran Balicer[3,14,16], Marinka Zitnik[1,3,17,18,19,†]

[1]Department of Biomedical Informatics, Harvard Medical School, Boston, MA
[2]Department of Engineering Science, University of Oxford, Oxford, UK
[3]The Ivan and Francesca Berkowitz Family Living Laboratory Collaboration at Harvard Medical School and Clalit Research Institute, Boston, MA, USA
[4]Cardiovascular Division, Department of Medicine, Brigham and Women's Hospital, Harvard Medical School, Boston, MA
[5]The Windreich Department of Artificial Intelligence and Human Health, Icahn School of Medicine at Mount Sinai, New York, NY, USA
[6]The Hasso Plattner Institute for Digital Health at Mount Sinai, Icahn School of Medicine at Mount Sinai and Mount Sinai Health System, New York City, NY, USA
[7]Mindich Child Health and Development Institute and the Departments of Pediatrics and Genetics & Genomic Sciences, Icahn School of Medicine at Mount Sinai, New York, NY, USA
[8]Mount Sinai Fuster Heart Hospital, Icahn School of Medicine at Mount Sinai, New York, NY, USA
[9]Mount Sinai AI Assurance Lab, Mount Sinai Health System, New York, NY, USA
[10]Institute for Critical Care Medicine, Icahn School of Medicine at Mount Sinai, New York, NY, USA
[11]Department of Medicine, Icahn School of Medicine at Mount Sinai, New York, NY, USA
[12]ATHENA-R1 Evaluation Group (the list of members and their affiliations appears in the Supplementary Information)
[13]Oxford Suzhou Centre for Advanced Research, University of Oxford, Suzhou, Jiangsu, China
[14]Clalit Research Institute, Innovation Division, Clalit Health Services, Ramat Gan, Israel
[15]Faculty of Computer and Information Science, Ben Gurion University of the Negev, Be'er Sheva, Israel
[16]Faculty of Health Sciences, School of Public Health, Ben Gurion University of the Negev, Be'er Sheva, Israel
[17]Kempner Institute for the Study of Natural and Artificial Intelligence, Harvard University, Cambridge, MA
[18]Broad Institute of MIT and Harvard, Cambridge, MA
[19]Harvard Data Science Initiative, Cambridge, MA

[†]Correspondence: marinka@hms.harvard.edu

1 Architecture, skills, and ATHENA-R1 inference

1.1 Overview

ATHENA-R1 is an AI agent for therapeutic reasoning that grounds its answers in medical evidence retrieved from a biomedical tool library through multi-step analysis. Given a treatment task in natural language, ATHENA-R1 produces a recommendation together with a reasoning trace that records each intermediate thought, the tool calls issued to retrieve evidence, and the evidence itself

(Figure 1). The trace makes the derivation of every conclusion inspectable. At each reasoning step, ATHENA-R1 decides what evidence is still needed, selects one or more tools from the library, issues tool calls to retrieve the evidence, and interprets the returned outputs in the context of the accumulated trace. It repeats this cycle until the accumulated evidence supports a final answer. The biomedical tool library contains 212 tools over FDA drug information, gene-disease evidence, phenotype ontologies, and target-disease associations (category distribution in Supplementary Figure 3). Two specialized tools support reasoning control: ToolRAG model, an embedding model that retrieves relevant tools from the library on demand, and Finish, which terminates reasoning and returns the final answer. ATHENA-R1's multi-step reasoning and tool-call behavior is obtained by fine-tuning an open-source LLM.

Training ATHENA-R1 requires the tool library and a corpus of therapeutic reasoning traces, neither of which can be produced at scale by hand. We therefore construct three multi-agent systems: ToolGen assembles the tool library from biomedical APIs, QuestionGen generates treatment tasks from verified sources, and TraceGen produces the reasoning traces that answer them. The three systems jointly yield the ATHENA-R1-Instruct dataset used to fine-tune ATHENA-R1; full details are given in Methods 3.

Preliminaries. During the inference process, given a therapy question Q, ATHENA-R1 generates a verified reasoning trace $\mathcal{R}_i = \{R_1, R_2, R_3, \ldots, R_i\}$ and the final answer A, which includes the rationale for the answer, such as treatment recommendations. The i-th step in the reasoning trace R_i consists of a thought T_i, a set of tool calls $C_i = \{C_i^1, C_i^2, \ldots, C_i^k\}$, and responses from the tool calls $\mathcal{E}_i = \{E_i^1, E_i^2, \ldots, E_i^k\}$, i.e., $R_i = \{T_i, C_i, \mathcal{E}_i\}$. The tool library $\mathcal{B} = \{B_1, B_2, \ldots, B_j\}$ contains a wide array of biomedical tools. $\mathcal{P}_i$ is the collection of tools available at step i, which contains the default specialized tools and tools retrieved by the ToolRAG model in previous steps. To aid understanding, Table 1 provides the complete definitions and explanations of all notations used in this work.

1.2 Skills of ATHENA-R1

The multi-step inference loop of ATHENA-R1 (Algorithm 1) composes five skills acquired by post-training the base LLM: generating the next reasoning thought, emitting tool-call arguments, deciding whether sufficient evidence has been gathered, retrieving new tools from the library, and summarizing long tool outputs.

Contextual thought generation. ATHENA-R1 is capable of generating thoughtful, context-aware, step-by-step reasoning based on prior interactions and user inputs. Given a user query Q and a sequence of previous reasoning traces $\mathcal{R}_{i-1}$, ATHENA-R1 produces a new thought T_i at step i, expressed in natural language. This thought generation process can be represented as:

$$T_i = \mathcal{F}_A(Q, \mathcal{R}_{i-1}, \mathcal{P}_i), \tag{1}$$

where $\mathcal{F}_A$ is the ATHENA-R1 backend LLM prompted by the ATHENA-R1 system prompt (the full system prompt for ATHENA-R1 and the agent prompts used by ToolGen, QuestionGen and TraceGen are listed in Supplementary Note 3), which operates in an autoregressive manner [1], and $\mathcal{P}_i$ is a set of available tools at this step. The query Q along with the reasoning traces $\mathcal{R}_{i-1}$ is provided as input to $\mathcal{F}_A$ to generate T_i, which incorporates reasoning about the analysis of prior steps and determines the next actions.

Tool call arguments generation. ATHENA-R1 executes tools by generating tool-call arguments conditioned on tool descriptions in the prompt. Each description specifies the tool's name, its purpose, and for each argument the name, purpose, data type, and whether it is mandatory (Supplementary Figure 2). Following the generated thought T_i of reasoning step R_i, given a set of tool descriptions $\mathcal{P}_i$ available to ATHENA-R1, ATHENA-R1 produces the corresponding tool call arguments:

$$C_i = \mathcal{F}_A(Q, \mathcal{R}_{i-1}, T_i, \mathcal{P}_i), \tag{2}$$

where $C_i = \{C_i^1, C_i^2, \ldots, C_i^k\}$ is a list that contains multiple tool calls as ATHENA-R1 supports parallel tool execution by generating multiple tool calls across various tools. The k-th tool call at step i, C_i^k, is represented as a code snippet in JSON format:

$$C_i^k = \left\langle \texttt{name}, \mathcal{A}_i^k \right\rangle, \tag{3}$$

where `name` is the name of the tool selected from the set of available tools $\mathcal{P}_i$, $\mathcal{A}_i^k = \{a_1, a_2, \ldots, a_n\}$ represents the arguments required by tool C_i^k, where each a_j corresponds to a specific argument-value pair corresponding to the description of the selected tool. The generated tool call arguments C_i are sent to the tool library codebase for execution. The results from these tool calls $\mathcal{E}_i = \{E_i^1, E_i^2, \ldots, E_i^k\}$, where E_i^k is the result of the k-th tool call, are then sent back to ATHENA-R1 as part of the current step of the reasoning trace $R_i = \{T_i, C_i, \mathcal{E}_i\}$. The reasoning trace $\mathcal{R}_i$ is then updated as: $\mathcal{R}_i = \mathcal{R}_{i-1} \cup R_i$. Tools used by ATHENA-R1 involve a variety of types, such as biomedical tools in tool library that gather outputs from multiple verified sources, and the machine learning-based tools

(ToolRAG model) that use a machine learning model to achieve certain functions. ATHENA-R1 is also registered as a callable tool in the library. When a parent ATHENA-R1 instance invokes it with a sub-question, a child ATHENA-R1 instance runs Algorithm 1 on the sub-question with full access to the tool library and ToolRAG model, and returns its final answer and reasoning trace to the parent as a standard tool response. This self-invocation supports the hierarchical decomposition used for complex therapeutic queries (Figure 1), in which the parent breaks the query into sub-problems, delegates them to child instances (which may run in parallel), and synthesizes their answers.

Logical multi-step reasoning and decision-making. ATHENA-R1 performs multi-step reasoning by alternating between thought generation and tool calls. At each step, the thought T_i assesses whether the accumulated trace, in particular the outputs of prior tool calls, contains enough evidence to answer the question. If it does, ATHENA-R1 issues the special Finish tool call to return the final answer A; otherwise it issues new tool calls C_i to retrieve additional evidence:

$$\begin{aligned} C_i &= \mathcal{F}_A(Q, \mathcal{R}_{i-1}, T_i, \mathcal{P}_i), \quad \textsc{Finish} \notin C_i; \\ A, C_F &= \mathcal{F}_A(Q, \mathcal{R}_{i-1}, T_i, \mathcal{P}_i), \quad \textsc{Finish} \in C_i, \end{aligned} \tag{4}$$

where C_F is a final tool call to the special Finish tool, signifying the end of the reasoning process. This iterative loop supports the hierarchical decomposition described above, in which child ATHENA-R1 instances handle delegated sub-problems.

Proactive tool search, selection, and use. The tool library contains 212 tools, too many to fit all descriptions into the LLM context window. ATHENA-R1 therefore retrieves tools on demand using ToolRAG model, an embedding model that maps each requirement to the top-k tools whose descriptions are most similar in embedding space. During inference, ToolRAG model precomputes embeddings for every tool description in the library; when ATHENA-R1 issues a ToolRAG model call with a requirement argument, ToolRAG model returns the top-k matches, which are added to the current tool set $\mathcal{P}_i$.

Retrieved tools are candidates, not commitments. Because ToolRAG model is imperfect and the reasoning is open-ended, not every retrieved tool fits the next action. ATHENA-R1 inspects the descriptions of tools in $\mathcal{P}_i$ against the prior trace, selects the ones that match its current need, and issues parallel tool calls to the selected tools.

Concise summarization. Tool outputs can often be lengthy. If retained verbatim, they restrict the maximum number of reasoning steps that fit in the context window. To address this, after each

reasoning step ATHENA-R1 rewrites the tool response $\mathcal{E}_i$ into a summary as follows:

$$\hat{\mathcal{E}}_i = \mathcal{F}_S\left(T_i, C_i, \mathcal{E}_i\right), \tag{5}$$

where $\mathcal{F}_S$ is the ATHENA-R1 backend LLM prompted to summarize. The summary keeps the content relevant to T_i so that longer reasoning traces fit within a fixed context budget.

Algorithm 1: ATHENA-R1 multi-step inference process.

Input: Question Q, tool library $\mathcal{B}$, Initial available tools $\mathcal{P}_0$
Output: Reasoning trace $\mathcal{R}$, final answer A

Initialize $\mathcal{R} \leftarrow \{\}$, tools $\mathcal{P} \leftarrow \mathcal{P}_0$, step $i \leftarrow 0$;
while *Reasoning is incomplete* **do**
  $i \leftarrow i + 1$;
  Generate thought: $T_i = \mathcal{F}_A(Q, \mathcal{R}_{i-1}, \mathcal{P}_i)$;
  Generate tool calls: $C_i = \mathcal{F}_A(Q, \mathcal{R}_{i-1}, T_i, \mathcal{P}_i)$;
  if *Finish in* C_i **then**
    Extract final answer A from Finish tool call;
    Return $\mathcal{R}_i, A$;
  else
    if *call to ToolRAG in* C_i **then**
      Execute ToolRAG and update $\mathcal{P}_i$;
    else if *call to ATHENA-R1 in* C_i **then**
      Spawn child ATHENA-R1 instance; child executes this algorithm on sub-question Q' from C_i;
      Collect child answer A' and trace $\mathcal{R}'$ as tool response $\mathcal{E}_i$;
    else
      Execute tools from C_i to get tool response $\mathcal{E}_i$;
    Update reasoning trace: $\mathcal{R}_i \leftarrow \mathcal{R}_{i-1} \cup \{T_i, C_i, \mathcal{E}_i\}$;

1.3 Core agentic abilities of ATHENA-R1

Composing the five skills above within the inference loop (Algorithm 1) yields four core agentic abilities that the Results evaluate: grounding answers in retrieved evidence, selecting tools based on the current reasoning goal, solving multi-step therapeutic problems, and retrieving information from continuously updated sources.

Grounding answers in retrieved evidence. Therapeutic decisions require answers that can be traced back to their source. Rather than generating a response from the LLM's parametric

knowledge, ATHENA-R1 issues tool calls to retrieve evidence from the tool library and composes the answer from the retrieved content. Supplementary Figure 1a illustrates this on a dosage question for Kisunla (donanemab-azbt), an FDA-approved drug from 2024 that postdates the base LLM's training data: ATHENA-R1 calls *get_dosage*, retrieves the dosage from the FDA label, and reports it in the final answer. The full reasoning trace makes each claim verifiable.

Goal-oriented tool selection. ATHENA-R1 selects tools based on the current reasoning goal rather than a fixed tool set. Supplementary Figure 1b shows ATHENA-R1 answering an adverse-reactions question for Alyftrek (vanzacaftor/tezacaftor/deutivacaftor): it issues a ToolRAG model call with a natural-language requirement, receives candidate tools, and selects *get_adverse_reactions* to retrieve the information from FDA drug labels. Because tool selection is driven by descriptions rather than identities, new tools added to the library at inference time are usable without retraining.

Multi-step therapeutic reasoning. Many treatment tasks cannot be answered by a single tool call: the answer depends on information that must first be resolved, combined, or re-queried when initial results are insufficient. ATHENA-R1 addresses such questions by chaining reasoning steps, each of which can issue new tool calls conditioned on the accumulated evidence. Supplementary Figure 1c shows this for identifying protein targets of breast cancer: ATHENA-R1 first calls *get_disease_id_desc* to resolve the EFO identifier, then calls *get_associated_targets* with that identifier to retrieve and rank the targets. No single tool in the library answers the question end-to-end; the two-step chain does.

Real-time retrieval from continually updated sources. Biomedical knowledge evolves: drug approvals, label changes, and new evidence appear continuously, while an LLM's parametric knowledge is fixed at the end of training. ATHENA-R1's tools query live APIs (openFDA, Open Targets, Human Phenotype Ontology), so updates to the underlying sources are available at inference time without retraining the LLM. This differs from document retrieval over precomputed embeddings, which requires re-indexing whenever the source corpus changes. Supplementary Figure 1d illustrates this for Bizengri (zenocutuzumab-zbco), approved by the FDA in December 2024, after the base LLM's training cutoff: ATHENA-R1 calls *get_indications*, which queries the openFDA API, and correctly reports the approved indications (non-small cell lung cancer and pancreatic adenocarcinoma).

2 Biomedical tool library

The tool library contains 212 biomedical tools built on APIs from openFDA [2], Open Targets [3], and the Human Phenotype Ontology [4], with complete category distribution shown in Supplementary Figure 3. The library covers adverse events, risks, and safety; addiction and abuse; drug patient populations; drug administration and handling; pharmacology; drug use, mechanism, and composition; ID and labeling; general clinical annotations; clinical laboratory information; general information for patients and relatives; disease-phenotype-target-drug links; biological annotation tools; publications; search; and target characterization. Each tool exposes a natural-language description that ATHENA-R1 uses to issue tool calls, and a backend that translates the call into an API request. Because the tools query live APIs at inference time, updates to the underlying sources are reflected without retraining or rebuilding any index. Manual construction of a library at this scale is impractical; we therefore built ToolGen, a multi-agent system that generates tool descriptions and the tool-to-API mappings automatically (§2.1).

2.1 ToolGen: a multi-agent system for constructing tools

ToolGen converts heterogeneous biomedical APIs into a unified set of tools callable by ATHENA-R1. The challenge is that source APIs use different schemas: Open Targets uses GraphQL, openFDA uses Elasticsearch, and the Human Phenotype Ontology uses REST. ToolGen comprises three GPT-4o agents (Extended Data Figure 1a) that jointly produce tools with a standard description format: a summarizer that enumerates the functions exposed by each API, a tool generator that produces a tool specification for each function, and a tool checker that validates the specification by generating and executing test calls. A final human-verification step reviews the validated tools before they enter the library.

API summarization. The summarizer ingests the API documentation and outputs a list of natural-language capabilities the API can support, such as "identify the active ingredients for a drug" or "find disease-related phenotypes". Each capability becomes a candidate tool.

Tool construction. For each capability in the list, the tool generator agent refers to the API documentation to create detailed tool specifications. These specifications include the tool's name, description, arguments, and specialized mapping data to translate arguments into API requests. Each argument includes the name, description, data type, and an indication of whether it is required. Mappings for Open Targets and the Human Phenotype Ontology correspond to the query string

defined in the GraphQL and RESTful schema, with variables in APIs connected to arguments in the tool. Mapping for openFDA uses the search and return fields of Elasticsearch, where search fields are tied to arguments in the tool, and return fields are selected to align with the tool's description (Supplementary Figure 2).

Tool check. The TOOL CHECKER agent evaluates the validity of generated tools by constructing and testing questions and tool calls. In this process, we first verify the mapping, ensuring its correctness by checking the validity of the provided mappings. Next, we randomly sample data points linked to either the input or output of the APIs, such as drug names, disease names, target names, and their corresponding IDs, to test the APIs. If useful information can be retrieved through API requests, the retrieved data and the tool specifications are sent to the TOOL CHECKER agent. The agent then generates test questions and tool call arguments for the tool. If the generated tool call arguments produce valid outputs, the tool is deemed functional and valid. However, if any of the steps in this process fail, the tool is marked invalid and subsequently removed.

Human verification. After the tool construction process, human experts manually verify and refine the tools. This evaluation includes determining whether the tool has meaningful applications, verifying that it functions as described, and ensuring its stability when handling unexpected inputs. Once this process is complete, the validated tools are included in the tool library.

3 Self-learning in ATHENA-R1

Training ATHENA-R1 to reason over hundreds of biomedical tools and across millions of drug, disease, and patient combinations is not feasible through human annotation. We therefore train ATHENA-R1 through two-level self-learning (Extended Data Figure 1 and Extended Data Figure 2). At the first level, multi-agent systems construct the tools, questions, and reasoning traces that form the ATHENA-R1-INSTRUCT dataset. An LLM is supervised fine-tuned on ATHENA-R1-INSTRUCT to produce an initial version of ATHENA-R1. At the second level, the supervised fine-tuned ATHENA-R1 is then refined through reinforcement learning against scientific feedback.

3.1 ATHENA-R1-INSTRUCT Data Sources

ATHENA-R1-INSTRUCT consists of three datasets generated by three agent systems. The tooling dataset contains augmented versions of 212 tools from tool library, with each tool description

rephrased to introduce variability so that ATHENA-R1 learns tool usage rather than memorizing specific descriptions. The treatment task dataset contains 85,340 tasks generated by QuestionGen from verified biomedical sources including FDA drug labels [2], PrimeKG [5], and Open Targets [3]. The reasoning trace dataset comprises 85,340 step-wise reasoning traces with 177,626 reasoning steps and 281,695 tool calls, generated by TraceGen. Processing these three datasets yields ATHENA-R1-Instruct with 378,027 instruction-tuning samples (Methods 3.6). The source information of ATHENA-R1-Instruct is collected from the following sources.

openFDA. openFDA is a health informatics database maintained by the FDA, offering public access to FDA data on approved drugs, devices, and foods [2]. This work utilizes the drug labeling data provided by the platform, which covers more than 67,000 drugs currently on the market [6]. openFDA includes a search API for retrieving information based on specific query fields. In this study, we use the drug API of openFDA to obtain FDA documentation on various drugs. Each drug entry contains numerous fields, such as indications, boxed warnings, and supply information (Table 1).

Open Targets. Open Targets is a platform that integrates data from 23 public resources, including Orphanet, Gene2Phenotype, and ChEMBL [3], to facilitate target identification and prioritization. As of September 2024, Open Targets includes 63,121 targets, 28,327 diseases, 18,041 drugs, 17,853,184 evidence entries, and 8,155,988 target-disease associations [7]. In this study, we utilize the association data from Open Targets to extract drug-disease relationships, drug status, drug-target interactions, and disease-target associations.

Human Phenotype Ontology (HPO). HPO is a database that provides an ontology of medically relevant phenotypes and disease-phenotype annotations [4]. It includes over 18,000 terms and more than 156,000 annotations linked to hereditary diseases. We leverage HPO to establish connections between diseases and phenotypes.

PrimeKG. PrimeKG is a comprehensive medicine-focused knowledge graph designed to offer a holistic view on diseases [5]. It incorporates data from 20 high-quality biomedical resources, capturing details about 17,080 diseases and their 4,050,249 relationships across ten key biological scales. In this work, PrimeKG provides the disease list and disease-related information for generating disease-related questions.

3.2 Tool graph

The tool graph is a directed graph that connects tools in tool library. It is used to facilitate the construction of training data. In the tool graph, each node represents a tool, and a directed link is established when the output of one tool serves as the input for another. The presence of a link is determined by providing descriptions of the two tools to an LLM, which then decides if a directed link should exist between them. Sampling a tool chain from a tool graph enables the construction of complex questions that require multiple rounds of tool calls. The tool graph is used solely for constructing the training dataset, not for the inference process in ATHENA-R1, due to the challenges in constructing an exceptionally precise tool graph. Unrestricted by the tool graph, ATHENA-R1 can seamlessly integrate newly added tools during the inference process.

3.3 QUESTIONGEN multi-agent system for question construction

While training ATHENA-R1 requires a large number of diverse questions that cover different aspects regarding treatment, disease, and drugs, and considers specialized cases such as patient populations, drug side effects, and drug interactions, manually writing these questions would be too costly. To effectively collect questions, QUESTIONGEN system is proposed as a question construction multi-agent system that generates meaningful questions from verified knowledge bases such as FDA documents (Extended Data Figure 1c). QUESTIONGEN system begins with the information extraction, which identifies and extracts key information relevant to the desired questions from documents and data sources. Using the extracted information, the question construction step creates questions, corresponding answers, and detailed explanations that clarify how the answer addresses the question. At last, the question evaluation step verifies questions in multiple aspects.

Question types. We generate questions through three distinct approaches: drug-centered, disease-centered, and tool-chain-centered question construction. Drug-centered questions focus on common therapeutic aspects of drugs, including their use in specific patient populations, indications, dosage, safety warnings, and potential risks. Disease-centered questions address specialized treatment scenarios. These questions incorporate detailed patient profiles, such as phenotypes, medical histories, current medications, and characteristics of the patient population. Tool-chain-centered questions are generated by randomly sampling a sequence of tools from the tool library tool graph, followed by creating questions based on the selected tool chain, which increases the diversity of questions.

Information extraction. This step identifies and extracts key information relevant to the desired questions from documents and data sources. Different information extraction strategies are designed to meet the requirements of various question types.

For drug-centered questions, we randomly sample drugs from the FDA database and retrieve their corresponding FDA documents as raw data sources. From each drug's FDA document, one field is randomly selected and extracted as the reference data for question construction. To enable question construction beyond just related to drug names, descriptive information about the drug, such as its mechanism of action, indications, and contraindications, is also extracted. This allows for the creation of questions that focus on the drug's characteristics without explicitly mentioning its name.

For disease-centered questions, we begin by randomly sampling a disease and gathering its description, associated phenotypes, targets, and all potential drugs. For each drug in the list, we retrieve its FDA document and extract information on indications, patient populations, contraindications, warnings, and drug interactions. The extracted data is then categorized by field and passed to the INFORMATION EXTRACTOR agent, which compares the drugs across these fields and highlights their differences. The generated comparison serves as the reference for question construction, enabling the creation of challenging, specialized questions that account for subtle differences among drugs.

For tool-chain-centered questions, we first sample a tool-chain from the tool graph starting from common tools such as identify drug ID or disease ID based on names. Then, we obtain information that can be retrieved by tools and the tool descriptions as the reference for question construction.

Question construction. In this step, the QUESTION GENERATOR leverages reference information extracted during the information extraction process to produce the question Q, corresponding answer G, and explanations justifying why the answers are correct X. For multiple-choice questions, it also generates answer options. The inclusion of explanations plays a crucial role, as they ensure the meaningfulness of the generated questions and offer solution hints for the HELPER agent during reasoning trace generation.

The QUESTION GENERATOR operates by prompting GPT-4o with instructions for generating questions. It utilizes multiple prompt variations, each designed for specific question types. During the question-construction process, general guidelines for question creation, reference information, and specific requirements for particular question types are provided as contextual input to the

QUESTION GENERATOR to produce the questions.

Question evaluation. The generated question undergoes evaluation based on three key aspects: knowledge-based grounding, answerability, and reasonableness. For each aspect, GPT-4o is prompted to perform the evaluation. For knowledge-based grounding, to ensure the question is generated from the reference information and not from hallucinations by the language model, both the question and reference information are provided to GPT-4o. GPT-4o is tasked with verifying whether the information in the question is directly derived from the reference information. For the answerability check, GPT-4o is prompted to assess whether the question can be adequately answered using the provided reference information. For the reasonableness check, the explanation in the generated question is sent to GPT-4o, which evaluates whether the reasoning behind the explanation is logical and makes sense. If any of the checks fail, the question is discarded. Otherwise, it is retained and sent to TRACEGEN for reasoning trace construction.

3.4 Reasoning trace generation

TRACEGEN is designed to generate training data consisting of a reasoning trace $\mathcal{R}$ and the final answer A based on the question Q. However, generating $\mathcal{R}$ faces several challenges: 1) Complexity of questions: Many questions require multi-step reasoning and analysis of multiple aspects, making it difficult to generate a single straightforward answer. The challenge is to create a reasoning trace that can handle these complexities effectively. 2) Incorporating external tools: To improve reasoning with the help of a massive number of tools, it is important to incorporate real-world tools into $\mathcal{R}$. The challenge here is integrating the outputs of these tools, rather than relying solely on the internal knowledge of LLMs. 3) Handling uncontrollable tool outputs: The results from external tools are often unpredictable. A key challenge is how to manage failure cases and continue progressing toward a solution, even when tool outputs deviate from expectations.

TRACEGEN is a multi-agent system designed to address various challenges through its key components: the HELPER agent, the TOOL PROVIDER module, and the SOLVER agent (Extended Data Figure 1d). The HELPER agent assists the SOLVER by offering step-by-step solution hints. It has access to the answers and explanations for questions and provides guidance for the next steps in the reasoning process based on the prior steps generated by the SOLVER agent. The TOOL PROVIDER presents a selection of potential tools for the SOLVER agent to choose from. These tools are identified based on reference information from the current question and a TOOLRAG model, which is iteratively trained on previously collected data to improve its recommendations. Armed

with tools from the Tool Provider module, hints from the Helper agent, the current question, and previously generated reasoning traces, the Solver agent iteratively solves the problem. It does so by generating subsequent reasoning steps and tool calls until arriving at the final answer.

Providing solution hint with Helper. The Helper agent plays a crucial role in assisting the Solver by providing solution hints, which is achieved by prompting GPT-4o with instructions. At each reasoning step i, the Helper has access to the problem question Q, the ground truth answer G, and its explanation X. Additionally, it takes as input the current reasoning trace $\mathcal{R}_i = \{R_1, R_2, \ldots, R_i\}$, which represents all steps generated by the Solver up to step i. Using this information, the Helper generates a solution hint, denoted as $\mathcal{H}_{i+1}$, which guides the Solver toward the next step in the reasoning process. When the Solver provides an answer A, the Helper checks whether A matches the ground truth answer G. If $A = G$, the reasoning process is deemed complete. If $A \neq G$, the Helper prompts the Solver to reflect on its reasoning and continue the reasoning process. In such cases, the Helper generates a hint $\mathcal{H}_{i+1}$ to guide the Solver back into the reasoning process and help refine the answer. Helper is defined as:

$$\mathcal{H}_{i+1} = \textsc{Helper}([Q, G, X], \mathcal{R}_i, A), \tag{6}$$

where A is empty if it is not provided to the Helper. By iteratively providing hints $\mathcal{H}_{i+1}$, the Helper ensures that the Solver progresses logically, generating the reasoning trace until the solution A is fully constructed and consistent with the ground truth answer G and explanation X.

Providing tools with the Tool Provider. The Tool Provider module supports the Solver by supplying relevant tools during the reasoning process. It operates in two stages. First, the module analyzes the reference information attached to the problem question Q and identifies an initial set of tools $\hat{\mathcal{P}}_0$ from the tool set $\mathcal{B}$. These initial tools are provided to the Solver at the start of the reasoning process. Second, if the Solver determines that no suitable tools are available for a specific reasoning step, it invokes the ToolRAG model within the Tool Provider module. This model retrieves additional tool suggestions, denoted as $\hat{\mathcal{P}}_i^{\text{RAG}}$, based on the tool descriptions provided by the Solver. By combining these two stages, the Tool Provider module ensures that the Solver has access to the most relevant tools throughout the reasoning process, either by leveraging the initial set of tools $\hat{\mathcal{P}}_0$ or dynamically adapting to the problem's demands with tools $\hat{\mathcal{P}}_i^{\text{RAG}}$ from the ToolRAG model.

Step-wise reasoning trace generation with Solver. The Solver serves as the central component for iteratively generating the reasoning trace $\mathcal{R}$ and deriving the final answer A, which

is achieved by prompting GPT-4o. We provide the algorithm in Algorithm 2. At each step i, the SOLVER integrates the question Q, tools from the TOOL PROVIDER, solution hints $\mathcal{H}_i$ provided by the HELPER, and its prior reasoning $\mathcal{R}_{i-1}$. Using this information, it formulates intermediate thoughts T_i and tool calls C_i, driving the reasoning process toward a complete and accurate solution. To simulate the inference process, the SOLVER avoids directly utilizing tools available in the initial set $\hat{\mathcal{P}}_0$. Instead, it generates virtual TOOLRAG calls, which simulate accessing these tools. Each virtual call specifies the tool's name and its rewritten description from $\hat{\mathcal{P}}_0$. These virtual calls are later replaced by actual calls to TOOLRAG. When the SOLVER identifies that no suitable tools are available for the current reasoning step, it invokes the TOOLRAG model within the TOOL PROVIDER to dynamically suggest additional tools. These new tools, denoted as $\hat{\mathcal{P}}_i^{\text{RAG}}$, are retrieved based on descriptions provided by the SOLVER. Hints $\mathcal{H}_{i+1}$ from the HELPER guide the SOLVER through the reasoning trajectory by suggesting logical next steps. This iterative mechanism allows the reasoning trace to evolve through external tool usage, dynamic adjustments based on feedback, and updates to the reasoning trace. The process continues until the FINISH tool suggests a candidate answer A. If validated by the HELPER, the reasoning trace $\mathcal{R}$ and the final answer A are returned. If the answer is deemed incorrect, the SOLVER removes the corresponding reasoning step and continues refining $\mathcal{R}$.

Reasoning trace evaluation. We consider the quality of the reasoning trace to be essential for the performance of ATHENA-R1. Evaluating the reasoning trace ensures both its reliability and correctness. This evaluation focuses on two main aspects: correctness and behavior. For correctness, we examine the correctness of the answer, reasoning trace, and tool calls. For answer correctness, in the case of multiple-choice questions, we compare the predicted option with the correct one. For open-ended reasoning questions, GPT-4 is prompted as a judge to determine if the prediction aligns with the correct answer. For the reasoning trace, we use GPT-4 as a judge, with the question and the ground truth answer serving as references to assess the quality of the reasoning process. For tool calls, we verify that the correct tool is used, and we check the correctness of the argument names, argument value types, and the inclusion of any required arguments. Even if the generated reasoning trace passes the correctness check, undesired behaviors may still occur, leading to incorrect reasoning during inference. In the behavior check, we examine issues such as hallucinations, arbitrary results, and repeated reasoning. For hallucinations, we look for hallucinated placeholders in object names and IDs, such as drug names, disease names, and target IDs in tool calls. Since IDs for drugs or diseases are not general knowledge, we eliminate reasoning traces where IDs appear without being shown earlier in the context. The goal of ATHENA-R1 is to generate verified reasoning traces, meaning the answers should be based on feedback from tool

Algorithm 2: Step-wise reasoning trace generation with SOLVER

Input: Question Q, tool library $\mathcal{B}$, ground truth G, explanation X
Output: Reasoning trace $\mathcal{R}$, final answer A

Initialize $\mathcal{R} \leftarrow \{\}$, tools $\mathcal{P} \leftarrow \{\}$, step $i \leftarrow 0$;
Obtain initial hints $\mathcal{H}_0$ from HELPER;
Obtain initial tools $\hat{\mathcal{P}}_0$ from TOOL PROVIDER;
while *Reasoning is incomplete* **do**
    $i \leftarrow i + 1$;
    if *Suitable tools exist in* $\mathcal{P}$ **then**
        Generate thought T_i and call C_i based on Q, $\mathcal{H}_i$, and $\mathcal{R}_{i-1}$;
    if *Suitable tools exist in* $\hat{\mathcal{P}}_0$ **then**
        Generate thought T_i and virtual calls C_i; **foreach** *Virtual call in* C_i **do**
            Replace virtual TOOLRAG call with real arguments;
    if *No suitable tools in* $\mathcal{P}$ **then**
        Generate thought T_i and request additional tools $\hat{\mathcal{P}}_i^{\text{RAG}}$ by calling TOOLRAG with desired tool's descriptions;
    Execute tool calls from C_i and update reasoning trace: $\mathcal{R} \leftarrow \mathcal{R} \cup \{R_i\}$;
    Obtain the next hint $\mathcal{H}_{i+1}$ from HELPER;
    if *FINISH tool provides candidate answer* A **then**
        if *HELPER confirms correctness* **then**
            Return $\mathcal{R}$, A;
        else
            Remove the R_i from $\mathcal{R}$;
Return $\mathcal{R}$, A;

calls. However, arbitrary results can arise when answers are derived from the model's unverified internal knowledge rather than tool feedback. We remove reasoning traces that are based on general knowledge instead of the feedback from the tools. In more complex cases, SOLVER may generate reasoning traces that include repeated thoughts or tool calls. For repeated thoughts, we assess the similarity between them, and for repeated tool calls, we identify steps with identical tool calls using the same arguments. We remove reasoning traces that involve repeated thoughts and tool calls. If the reasoning trace passes all checks, it is retained. However, if it fails due to errors in correctness or undesired behaviors, it is discarded. This evaluation ensures that only high-quality reasoning traces are considered in training data.

3.5 Iterative training for TOOLRAG model

TOOLRAG model is used in both ATHENA-R1's inference process and the training data collection phase. It utilizes gte-Qwen2-1.5B-instruct [8] as the base model, which is fine-tuned on pairs of requirements and tool descriptions using the multiple negatives ranking loss. In the TOOL PROVIDER module, we use the TOOLRAG model to identify tools beyond the initial list retrieved from the reference information of the question. While the TOOLRAG model requires training data from reasoning traces, we propose an iterative training process for TOOLRAG model, where it is trained on the generated reasoning traces, which in turn helps improve the generation of future reasoning traces. In the first stage, since the TOOLRAG model is not yet available, we rely solely on the initial set of tools $\hat{\mathcal{P}}_0$ that are obtained from the reference information of the question, to generate the reasoning trace. From this trace, we extract pairs of tool requirements and tool descriptions, which are then used to train the TOOLRAG model. In the second stage, after the initial training of TOOLRAG model, we use it to select tools instead of relying exclusively on $\hat{\mathcal{P}}_0$. This approach allows the reasoning trace to better reflect real-world use cases, as tools are now retrieved directly by the TOOLRAG model. Using the data collected from this stage, we continue to gather new pairs for further training of the TOOLRAG model. This process is repeated iteratively, continually refining both the TOOLRAG model and the quality of reasoning trace generation.

3.6 ATHENA-R1 level-1 training: Data preparation and augmentation

To enable the multi-step reasoning and tool call capabilities of ATHENA-R1, we fine-tune LLMs using the ATHENA-R1-INSTRUCT dataset designed to encompass the diverse behaviors required by ATHENA-R1. This subsection introduces the training dataset ATHENA-R1-INSTRUCT and the training strategies.

ATHENA-R1 Training Dataset: ATHENA-R1-INSTRUCT dataset. We use three agent systems to generate three training datasets, including a tooling dataset, a treatment task dataset, and a reasoning trace dataset. The treatment task dataset comprises 85,340 treatment tasks, while the reasoning trace dataset includes 177,626 reasoning steps and 281,695 tool calls. Then, we begin by integrating the question with a reasoning trace and incorporating augmented tools. Next, we break down the complete reasoning trace into step-wise training data. This process results in the creation of the ATHENA-R1-INSTRUCT dataset that contains 378,027 instruction tuning data samples. The ATHENA-R1-INSTRUCT dataset is generated by randomly sampling from drugs in the FDA drug label database and disease phenotypes from the PrimeKG database. To prevent any leakage of

evaluation data through the training data, we remove all drugs approved after 2023.

Constructing step-wise training data. During supervised fine-tuning, in order to enable ATHENA-R1 to have step-wise reasoning and tool call abilities, we apply step-wise supervision on the thoughts and tool calls at each reasoning step. Given a question Q, a reasoning trace $\mathcal{R} = \{R_1, R_2, R_3, \ldots, R_M\}$ consisting of M reasoning steps, and the final answer A, where each reasoning step R_i is represented as a tuple $R_i = \{T_i, C_i, \mathcal{E}_i\}$ (with T_i and C_i being the thought and tool calls at the i-th step, and $\mathcal{E}_i$ results of tool calls), we decompose the reasoning trace into M step-wise samples for supervision. Each of these step-wise samples consists of an input and an output for the fine-tuned LLM.

For each $i \in \{1, 2, \ldots, M-1\}$, the input to the model consists of the system prompt S, question Q, a set of available tools at step i denoted as $\mathcal{P}_i$, and the reasoning trace up to the previous step, denoted as $\mathcal{R}_{i-1}$, which represents the reasoning steps from R_1 to R_{i-1}. The output of the model is the components $[T_i, C_i]$, which correspond to the thought and tool calls at step i. At the final step M, the input consists of the system prompt S, question Q and the reasoning trace up to the $M-1$-th step, i.e., $\mathcal{R}_{M-1} = \{R_1, R_2, \ldots, R_{M-1}\}$, as well as the tools available up to that step, $\mathcal{P}_M$. The output consists of the thought T_M, the tool calls C_M at step M (containing the FINISH call C_F), and the final answer A. Thus, for each $i \in \{1, 2, \ldots, M\}$, the i-th step-wise sample is:

$$\begin{aligned} &\text{Input: } [S, Q, \mathcal{R}_{i-1}, \mathcal{P}_i], \quad \text{Output: } [T_i, C_i] \quad \text{for} \quad i \in \{1, 2, \ldots, M-1\}, \\ &\text{Input: } [S, Q, \mathcal{R}_{M-1}, \mathcal{P}_M], \quad \text{Output: } [T_M, C_M, A] \quad \text{for} \quad i = M. \end{aligned} \tag{7}$$

While each step in the reasoning trace may involve multiple tool calls, we introduce an argument, ID, which is a randomly generated string, to uniquely identify each tool call. This ID is added to both the tool call arguments $C_{i,k}$, and the corresponding results returned by the tool $\mathcal{E}_{i,k}$. The ID is added to the input reasoning trace $\mathcal{R}_{M-1}$ and is removed from the model output, as it is a random, unpredictable string.

This step-wise decomposition allows for effective supervision of the reasoning process, with each sample providing contextual information about the model's reasoning at every intermediate step. At the final step, both the reasoning components and the final answer are output together, marking the completion of the reasoning process.

Training data augmentation. We design several training data augmentation strategies to ensure that ATHENA-R1 is trained to perform tool calls based on contextual information and can generalize to new tools.

Augmenting tools. To prevent over-fitting to the tools in tool library, we apply augmentation to the tool descriptions by randomly rephrasing all fields of a tool. For each tool in tool library, we prompt the LLM to rewrite the original description and generate 20 distinct versions of the tool's name, description, argument names, and argument descriptions. For each training sample in ATHENA-R1-INSTRUCT, we randomly select from these rewritten fields to create a new, augmented tool description. Then, we replace the tool name and argument names in the tool call arguments to generate the augmented training sample. This strategy enables ATHENA-R1 to learn how to call tools based on the tool names and arguments, rather than memorizing the specific tools encountered during training. As a result, the model can generalize to new and unseen tools during inference, enabling flexible scaling of tool library.

Extending the available tool set. The available tool set $\mathcal{P}$ (index note of i-th step is omitted here for simplicity) in each sample of ATHENA-R1-INSTRUCT consists of tools used in the reasoning traces. However, during inference, the tools retrieved by the imperfect TOOLRAG model may include additional candidates, making it challenging for ATHENA-R1 to select the most suitable tools from the returned set. To address this, we enhance the tool set $\mathcal{P}$ by including all tools retrieved by TOOLRAG model, not just those explicitly used in the reasoning traces. Additionally, we randomly sample several tools from tool library and add them to $\mathcal{P}$. This approach ensures that ATHENA-R1 learns to effectively select the correct tool from a broader set of candidate tools.

Shuffling the tool list. To mitigate any potential bias introduced by the position of tools in the tool list $\mathcal{P}$, we shuffle the tools in $\mathcal{P}$. This ensures that the order in which tools appear does not influence the ability of ATHENA-R1 to select the correct tool. By randomizing the positions of the tools, we encourage ATHENA-R1 to focus on the context and descriptions of the tools, rather than their position in the list. This strategy helps the model learn to make tool selections based on the contextual information rather than relying on the order of the tools in the tool set.

Replacing long results with summaries. To ensure the training data fits within the context window while preserving the overall reasoning trace, we shorten samples that exceed the maximum context limit by replacing the full tool results with summarized versions. This process begins with the earliest step in the reasoning trace and continues until the total length of the sample is within the context limit.

3.7 ATHENA-R1 level-1 training: Training design

Model. We use pre-trained LLMs as the base models for fine-tuning on the ATHENA-R1-Instruct. The base model is built on the Transformer architecture [9], which leverages self-attention mechanisms to process input sequences in parallel. This enables efficient learning of contextual relationships between tokens. Our model initialization begins with loading the pre-trained weights from the instruction-tuned LLM. To further adapt the model to our specific task, we apply Low-Rank Adaptation (LoRA) fine-tuning [10]. LoRA enhances the fine-tuning process by introducing low-rank updates to the pre-trained weights, reducing computational costs and the number of parameters trained. This allows us to efficiently fine-tune the model while preserving the knowledge from the pre-trained LLM.

Training process. During the training process, the input of one training sample $[S, Q, \mathcal{R}_{i-1}, \mathcal{P}_i]$ and the corresponding output $[T_i, C_i]$ are formatted according to the instruction-following format of the LLM (e.g., the chat template of the base model). The formatted text is then processed by the LLM tokenizer to generate a sequence of tokens $\mathbf{x} = \{x_1, x_2, \ldots, x_N\}$, where N is the total number of tokens. These tokens are embedded and passed through the model for autoregressive prediction. The model predicts the next token x_t based on all previously generated tokens $\{x_1, x_2, \ldots, x_{t-1}\}$, producing a conditional probability distribution $p(x_t \mid x_1, x_2, \ldots, x_{t-1})$. The training objective is to minimize the autoregressive loss, but only for the tokens corresponding to the output sequence $\mathbf{x} \subseteq \mathbf{x}_{\text{out}}$. The loss is defined as:

$$\mathcal{L} = -\sum_{t \in \text{Idx}_{\text{out}}} \log p(x_t \mid x_1, x_2, \ldots, x_{t-1}), \tag{8}$$

where Idx_{out} represents the indices of tokens in $\mathbf{x}$ corresponding to the output sequence $\mathbf{x}_{\text{out}}$. By focusing on the output tokens, this loss ensures that the model learns to generate thought and tool calls instead of over-fitting to the results from tools.

Training implementation. Training resources. We use the Nvidia H100 GPU cluster provided by the Kempner Institute for the Study of Natural and Artificial Intelligence at Harvard University to train ATHENA-R1. For training the ATHENA-R1-8B model, we utilize 4 GPUs, totaling 320GB of GPU memory. Training ATHENA-R1-8B requires 9.93 GPU days.

Training infrastructure. The training infrastructure of ATHENA-R1 is modified based on several key libraries, including TRL [11], Alignment Handbook [12], Transformers [13], Deepspeed [14],

and PyTorch [15]. The fully sharded data parallel (FSDP) technique is employed as the multi-GPU distributed training method. In this setup, the model's parameters are split across multiple GPUs to reduce memory usage, enabling the training of ATHENA-R1 with large backend LLMs and long context windows. This approach distributes both computation and model weights. For multi-node training, we use PyTorch implementation of FSDP. For single-node training, the Deepspeed implementation of FSDP is utilized, which is also referred to as Deepspeed Stage 3.

3.8 ATHENA-R1 level-2 training: Reinforcement learning

After supervised fine-tuning, we refine ATHENA-R1's policy through reinforcement learning (RL) in which ATHENA-R1 explores its environment of 212 biomedical tools and receives scientific feedback on its reasoning traces (Extended Data Figure 2). This stage uses no new human labels: the reward is computed entirely from rule-based checks against the ground-truth answer, tool specifications, and the structure of the reasoning trace.

Multi-turn rollout environment. During RL training, ATHENA-R1 samples questions from the ATHENA-R1-Instruct dataset and generates reasoning traces by interacting with the same tool library used at inference. For each training prompt, the model produces n=5 independent rollouts at temperature T=1.0. Each rollout proceeds in two stages. In the reasoning stage, the model alternates between generating a thought with one or more tool calls, executing those calls against tool library, and receiving the tool outputs as context. Invoking the Finish tool transitions the rollout to the mapping stage, in which the model is shown the multiple-choice options for the question and must select one. The rollout terminates when the mapping stage produces a valid answer.

Several safeguards prevent non-terminating rollouts: the mapping stage is allowed one retry before termination; two consecutive tool-call errors or empty assistant turns in the reasoning stage end the rollout; and a hard cap of 30 reasoning turns applies. Single tool-call failures do not terminate the rollout: an error message is returned as a tool-role response and the model continues reasoning, with invalid calls penalized through the reward function. When the accumulated conversation approaches the 16,384-token response budget, an in-rollout summary mechanism compresses context before the next generation. At the first level, the oldest tool-output messages exceeding 3,000 characters are replaced, one at a time, with concise summaries produced by the current policy using a fixed summarization prompt; compression stops once the token count falls below the budget. If the first level is insufficient, a second level summarizes the full prior conversation in chunks. The rollout then continues from the current turn with the compressed context in place of the original outputs.

Design of rewards for RL. Each rollout is scored by a composite reward of 12 rule-based checks grouped into six dimensions (Table 1):

1. **Answer correctness** (1 check): Exact match of the extracted answer letter (A–E) against the ground-truth option. This is the dominant learning signal.
2. **Output-format validity** (3 checks): Tool-call blocks must be parseable JSON; the final response must follow proper thought-answer structure; and tool names and argument keys must match tool specifications.
3. **Evidence gathering** (1 check): The model must call TooLRAG model before invoking domain tools, ensuring systematic tool retrieval rather than guessing tool names.
4. **Multi-step reasoning** (2 checks): At least two distinct tool-call steps must be executed, and the response length must fall within a calibrated range.
5. **Tool-argument grounding** (2 checks): Identifiers used in tool arguments (drug IDs, disease IDs) must originate from the question or prior tool outputs, not fabricated; no placeholder strings such as "CHEMBLXXXX" may appear in arguments.
6. **Reasoning non-redundancy** (3 checks): No identical sentences repeated across thought blocks; no exact-duplicate tool calls or excessive calls to the same tool; no structural anomalies such as multiple closing tags in a single message.

The per-rollout reward is the weighted sum of all 12 checks. Answer correctness (weight 5.0) dominates the gradient signal; structural terms (weights 0.5–1.0) act as gating constraints that shape well-formed traces without overwhelming the primary accuracy signal. To reinforce the evidence-gathering strategy, rollouts that skip TooLRAG model (i.e., the evidence-gathering check scores 0) incur a compound penalty: six other structural checks, marked with † in Table 1, are additionally scaled to 80% of their listed weight. Skipping TooLRAG model therefore costs both the evidence-gathering reward itself and 20% of six structural-quality rewards.

GRPO policy optimization. The policy is updated via group relative policy optimization (GRPO) [16], which computes advantages from the relative reward within each group of n=5 rollouts drawn from the same prompt, removing the need for a learned value function or critic. For each prompt q with rollouts $\{\tau_1, \ldots, \tau_n\}$ receiving rewards $\{r_1, \ldots, r_n\}$, the advantage for rollout j is the group-normalized reward:

$$\hat{A}_j = \frac{r_j - \bar{r}}{\sigma_r + \epsilon}, \qquad \bar{r} = \frac{1}{n}\sum_{k=1}^{n} r_k, \qquad \sigma_r = \sqrt{\frac{1}{n}\sum_{k=1}^{n}(r_k - \bar{r})^2}, \tag{9}$$

Table 1: Scientific feedback reward: 12 checks grouped into six dimensions. Checks marked with † are conditionally scaled to 80% of their listed weight when the evidence-gathering check is not satisfied.

Dimension	Check	Rule	Weight
Answer correctness	Accuracy	Extracted letter matches ground truth	5.0
Output-format validity	Tool Call Format	`<tool_call>` blocks are parseable JSON	0.5
	Final Format	Proper thought and answer structure	0.5
	Tool Call Validation†	Tool names and argument keys match tool specs	0.5
Evidence gathering	ToolRAG-first	ToolRAG model called before domain tools	1.0
Multi-step reasoning	Minimum tool-call steps	At least 2 distinct tool-call steps	1.0
	Answer length bound	Response length in 120–260 words	0.8
Tool-argument grounding	Identifier provenance†	Identifiers appear in question or prior outputs	0.5
	No placeholder identifiers†	No placeholder strings in arguments	0.5
Reasoning non-redundancy	Thought non-repetition†	No identical sentences across thought blocks	0.5
	Tool-call non-repetition†	No duplicate tool calls; at most 10 to same tool	1.0
	Thought boundary integrity†	No structural anomalies in thought boundaries	1.0

where ϵ is a small constant for numerical stability. The policy is trained with the geometric-mean variant of the GRPO objective [17], which replaces GRPO's arithmetic mean of token-level rewards with the geometric mean. This reduces the influence of outlier tokens on policy updates and keeps importance-sampling ratios within a more stable range. A frozen copy of the SFT checkpoint serves as the reference policy, against which we apply a KL divergence penalty with coefficient 0.01. Because the reference is fixed throughout training, the KL penalty anchors the RL-refined policy to the SFT distribution, preserving its capabilities while allowing reward-driven improvements. Training is purely on-policy with no replay buffer: each gradient step generates fresh rollouts, scores them, computes advantages, and updates the policy. The full training procedure is given in Algorithm 3.

Algorithm 3: GRPO training for ATHENA-R1 with multi-turn tool interaction

Input: SFT checkpoint θ_0, prompts $\mathcal{D}$ from ATHENA-R1-INSTRUCT, tool library $\mathcal{B}$, reward function ρ, rollouts per prompt n=5

Output: Refined policy π_θ

```
Initialize policy π_θ ← θ_0, frozen reference π_ref ← θ_0;
for each training step do
    Sample batch {q_1, ..., q_B} from D, B=512;
    for each prompt q_i do
        for j = 1 to n do
            Initialize rollout τ_j ← {};
            repeat
                Sample thought T and tool calls C from π_θ;
                Execute C against tool library B;
                Append (T, C, results) to τ_j;
            until mapping stage produces valid answer, or safeguard triggers;
        Score rollouts: r_j = ρ(τ_j, G_i) for j = 1, ..., n;
        Compute group-normalized advantages: Â_j = (r_j − r̄)/(σ_r + ε);
    Update θ via clipped GRPO objective with KL penalty against π_ref;
Return π_θ;
```

Training configuration. We train with a batch size of B=512 prompts per gradient step (2,560 rollouts total). The clipping ratio is 0.4, and the learning rate is 5×10^{-7}. Training is conducted using the verl framework [18] on NVIDIA H100 GPUs.

4 Model benchmarking

4.1 Benchmark datasets

We constructed five evaluation benchmarks, including DrugPC, BrandPC, GenericPC, DescriptionPC, and TreatmentPC. Given that LLMs have been trained on publicly available data, there is a risk of potential data leakage, meaning that LLMs may have previously encountered similar questions.

To mitigate this risk in our evaluation datasets, we focused on creating new datasets centered around drugs approved by the FDA in 2024, reducing the likelihood that the LLMs have been

exposed to this specific information. To further assess benchmark quality independently of the construction pipeline, we report human expert agreement rates on a subset of questions and note that all benchmark drugs postdate the training data cutoff, making parametric recall unlikely to explain performance differences. The construction workflow for initial question generation, combined with human expert review and open-ended evaluation rather than multiple-choice format alone, reduces the specific bias modes associated with each approach individually. Statistics of all benchmarks are shown in Table 1.

DrugPC: A comprehensive benchmark covering 11 common therapeutic tasks. We created the DrugPC dataset, which includes 3,168 questions covering 11 common tasks related to therapy. These sub-tasks include drug overview, drug ingredients, drug warnings and safety, drug dependence and abuse, dosage and administration, use in specific populations, pharmacology, clinical information, nonclinical toxicology, patient-focused information, and storage and supply (as detailed in Table 1). To facilitate evaluation, the dataset is formatted as multiple-choice questions, with each question followed by several options (most having 4 options, with some having 2 or 5). The dataset construction process follows these steps: 1) We classify the sections within FDA documents and map them to 11 tasks. The specific fields for each task are outlined in Table 1. 2) For question construction, we use the text from relevant sections of FDA documents as context. Using the question construction multi-agent system QUESTIONGEN, we create questions, multiple-choice options, and corresponding answers that can be answered using the provided context. The evaluation process checks whether the questions are answerable based on the context provided and ensures that the answers are accurate according to the given information. 3) After construction, a human evaluation process is conducted to carefully review and refine the questions and answers, ensuring that non-biomedical content, such as information about drug manufacturers, is excluded.

BrandPC/GenericPC: Datasets representing drug name in brand and generic forms. LLM-based methods have been shown to be sensitive to variations, such as representing drugs by either their brand or generic names. To assess the robustness of ATHENA-R1, we transform the DrugPC dataset into two versions: BrandPC and GenericPC. In these versions, drug names are systematically replaced with their respective brand or generic names. Problems that do not involve drug names in the questions or options remain unchanged, while those requiring conversion between brand and generic names are also kept as is.

DescriptionPC: A benchmark representing drugs with detailed descriptions. The drug name plays a crucial role in enabling LLM-based methods to effectively answer questions.

However, to evaluate the models' generalization capabilities in the absence of explicit drug names, we introduce the DescriptionPC benchmark. In this benchmark, drug names are replaced with detailed descriptions that include information such as indications, mechanisms of action, contraindications, and drug interactions. To ensure the validity of the dataset, we manually remove questions in the DrugPC benchmark that cannot be answered after replacing the drug name with its description. This process results in 626 questions, forming the DescriptionPC benchmark. While a model might infer an answer without explicitly identifying the drug name from its description, ensuring that the prediction is based on the correct drug rather than exploiting patterns is critical. To address this, the DescriptionPC benchmark incorporates a two-step evaluation process: drug identification and answer correctness evaluation. **Drug identification**: The model must identify the drug name based on the provided description. Since multiple drugs can share similar descriptions, we construct the ground truth for this step by first collecting drug descriptions corresponding to their original names. We then identify similar drugs that can be described in the same way and include them in the ground truth. **Answer correctness evaluation**: Using the drug names predicted in the first step, the model is tasked with selecting the correct answer from multiple-choice questions. During the two-step evaluation process, if the drug identification in the first step is incorrect, the second step is automatically marked as incorrect, regardless of the answer's correctness in that step. This approach ensures that the evaluation rigorously tests the model's reasoning based on the intended drug descriptions.

TreatmentPC: A specialized treatment benchmark for precision therapy in targeted conditions. While multiple indications can be applied to a single disease, patients with specific conditions, such as pregnancy or comorbidities, require specialized treatment approaches, such as customized drug selection and dosage adjustments. The TreatmentPC benchmark is designed to address such specialized treatment scenarios by generating questions based on the varying application conditions of drugs. This is achieved using the question construction system QuestionGen. We first select drugs approved by the FDA in 2024, identifying their indicated diseases. For each disease, we compile all associated treatments and analyze the unique attributes of each drug. This analysis is conducted by examining FDA documents, including information on indications, usage in specific populations, safety warnings, precautions, and contraindications. Next, we generate multiple-choice questions that specifically account for differences among drugs. The answer options represent treatments for the disease, but only one is suitable based on the patient's specific condition. For instance, we include scenarios where a patient is taking other medications that are contraindicated for certain treatments. The TreatmentPC benchmark requires the model to do a thorough analysis of

the patient's condition before determining an appropriate solution.

4.2 Evaluation strategy

Multiple-choice evaluation. In this approach, the question is accompanied by multiple options, and the model must select the correct answer from these options. Accuracy across the dataset is reported as the evaluation metric. Examples of multiple-choice treatment questions are in Table 2.

Open-ended evaluation. The model is presented with only the question, without any options, and is required to generate an open-ended answer. Evaluating such answers is inherently challenging. To address this, we introduce an additional step: the generated open-ended answer is provided as context, and GPT-5 is used to select the correct answer from multiple options based on this context. This allows for the evaluation of open-ended questions by producing quantitative results. For both multiple-choice evaluation and open-ended evaluation, we report the accuracy on the benchmark dataset as the performance metric. Examples of open-ended treatment questions are in Table 2.

Real-world evaluation settings. We evaluate ATHENA-R1 through three real-world studies. The first is a cross-organizational expert assessment in which biomedical experts from 28 disease organizations rate ATHENA-R1's responses in blinded head-to-head comparisons against reference models (§5). The second is a clinical case-based expert review in which three physicians rate ATHENA-R1's reasoning on clinical vignettes constructed from real hospitalized patients (§6). The third is a population-scale validation in which ATHENA-R1-generated adverse-event predictions are tested against longitudinal electronic health records from over 5 million patients (§7). Together, these three arms test ATHENA-R1 at three scales: cohort-level expert judgment, case-level clinician review, and population-level outcome validation, on evidence that is independent of the training and benchmark data.

5 Real-world evaluation: Rare disease treatment reasoning

We evaluated ATHENA-R1 in therapeutic reasoning scenarios through a blinded, arena-based comparison against reference models, rated by human disease experts. Participant demographics, complete Likert anchor text for each criterion, per-criterion statistical tables, and inter-rater agreement

details are reported in Supplementary Note 4.

5.1 Evaluator recruitment, expertise, and consent

We recruited evaluators from two sources. First, 29 experts from 28 disease organizations (Supplementary Table 4) joined through a collaborative research program in which each organization contributed expertise in its disease area. Each organization designated one or more biomedical experts as evaluators, with the number set by the organization's capacity and the breadth of its disease focus. The organizations spanned neurodevelopmental and epileptic disorders, metabolic diseases, rare cancers, channelopathies, and autoimmune and inflammatory conditions, and their work covered drug development from target identification and preclinical testing through clinical trials and approved therapies. Twenty-seven of the 28 organizations were based in the United States and one in Mexico. Second, 9 biomedical researchers from academic institutions joined as evaluators, all with verified life-sciences credentials.

Across all recruited evaluators, self-reported roles included organization lead (17), non-clinical researcher (6), clinical researcher (4), clinician (3), and additional roles such as chief scientific officer and scientific director. Professional backgrounds included PhD researchers or scientists (10), board-certified physicians (3), clinical researchers (4), and patients, caregivers, or patient advocates (17); several evaluators reported more than one role.

Of the 29 organizational experts, 14 completed blinded evaluations within the study period and contributed 94 responses; the remaining 15 did not complete evaluations because of scheduling constraints. Combined with 16 responses from the 9 additional evaluators, this yielded 110 expert-evaluated responses from 23 evaluators in total.

The evaluation used model-generated outputs only. No patient data, protected health information, or clinical records were involved at any stage, and the study was exempt from Institutional Review Board review. All participants provided informed consent for their anonymized ratings to be reported in aggregate.

5.2 Question construction from organization-specific context

Treatment tasks for the evaluation were generated with the QuestionGen multi-agent system (§3.3, Extended Data Figure 1b), using disease-specific context supplied by each participating organization as reference input in place of the randomly sampled drugs and diseases used during training-data

generation. Each organization completed a structured intake form specifying (1) the disease area of focus, (2) relevant medical specialties, (3) drug-development stage (*e.g.* preclinical, clinical trial, post-market), and (4) clinical endpoints or treatment aspects of interest. QuestionGen used this information to construct drug-centered and disease-centered questions addressing indications, dosage, safety warnings, drug interactions, detailed patient profiles, and population-specific treatment constraints. Generated questions passed the three-step verification pipeline (knowledge-based grounding, answerability, reasonableness) described in §3.3. The evaluation question set was finalized from organization-supplied context before any model produced a response, and no question was added, removed, or revised on the basis of how ATHENA-R1 or any reference model answered it; question construction was therefore blind to model outputs. Questions were presented in open-ended format with no predefined answer choices, requiring each model to produce a free-form therapeutic response together with its full reasoning trace; this format reflects realistic clinical reasoning scenarios in which practitioners must synthesize information without pre-specified options. Because organization-specific contexts were collected after ATHENA-R1-Instruct construction and never entered training, the evaluation question set is independent of the ATHENA-R1-Instruct training data.

5.3 Evaluation interface, criteria, and protocol

Evaluations were conducted through a web-based arena interface designed for blinded, side-by-side comparison (Figure 3b). The treatment task was displayed at the top of the screen, with two model responses, one from ATHENA-R1 and one from a reference model, shown below it side by side and labeled "Model A" and "Model B"; model identities were hidden from evaluators throughout. The primary reference model was Qwen3-8B ($n = 100$), the base model from which ATHENA-R1 is built; comparing ATHENA-R1 against its own unmodified base at the same parameter scale isolates the contribution of ATHENA-R1's tool library and self-learning while holding the underlying language model fixed. The remaining responses spot-checked ATHENA-R1 against frontier and different-family systems: o3-mini ($n = 3$), Gemini-2.0-Flash ($n = 3$), DeepSeek-R1 ($n = 2$), DeepSeek-R1-Distill-Llama-8B ($n = 1$), and Llama-3.1-8B-Instruct ($n = 1$), where n is the number of evaluated responses. The assignment of ATHENA-R1 and the reference model to positions A and B was randomized independently per question to prevent position bias. Each evaluator received a set of questions matched to their organization's disease area, with question order randomized independently per evaluator. Evaluators were instructed to read each treatment task carefully, review both model responses in full, and then provide their ratings and preferences. No time limit was

imposed on individual questions, and evaluators could complete their assigned set across multiple sessions.

For each response, evaluators scored both models on a 1–5 Likert scale across eight criteria and selected a pairwise preference for each of the same criteria. The criteria capture distinct aspects of therapeutic response quality: *task success* (whether the requested task is completed), *helpfulness of rationale* (how clearly the reasoning explains the derivation of the answer), *cognitive traceability* (how easy the chain of reasoning is to follow), *possibility of harm* (whether acting on the output could cause harm in a clinical or research setting), *alignment with clinical consensus* (agreement with established clinical guidelines and best practices), *accuracy of content* (factual accuracy of the information provided), *completeness* (coverage of relevant aspects of the question), and *clinical relevance* (applicability to real-world clinical or research practice). Full Likert anchor descriptions for each criterion are provided in Supplementary Note 4. An "Unable to Judge" option was available for any absolute rating and was excluded from per-criterion means. For pairwise preference, evaluators chose one of four options: "Model A is better," "Model B is better," "Both are equally good," or "Neither did well."

5.4 Statistical analyses and inter-rater agreement

Pairwise preferences were compared against a null of 50% using a one-sided binomial test on decisive comparisons (excluding "Both are equally good" and "Neither did well"), testing whether ATHENA-R1 was preferred more often than the reference model. Absolute ratings were compared between ATHENA-R1 and reference models using a one-sided Wilcoxon signed-rank test on paired scores per question, testing whether ATHENA-R1 scores are systematically higher, with "Unable to Judge" ratings excluded pairwise. Descriptive statistics are reported as mean ± standard deviation (Figure 3d); effect sizes for the Wilcoxon signed-rank test are reported as rank-biserial correlation r (Supplementary Note 4, Table 3). A significance threshold of $P < 5 \times 10^{-5}$ was applied uniformly across all eight criteria.

Inter-rater agreement was computed on the seven questions independently rated by multiple evaluators (six questions by two evaluators and one by three evaluators, yielding nine unique evaluator pairs). Overlap arose incidentally from evaluator assignment by disease area rather than by design. For pairwise preference, agreement was defined as both evaluators selecting the same one of the four options after normalizing for randomized A/B position assignment. For absolute ratings, we report exact agreement (identical Likert scores), within-one-point agreement, and mean absolute

score difference per evaluator pair. Overlap questions were not aggregated across evaluators; each evaluator-response rating is reported as an independent observation in the main analyses. Because the number of overlap questions is small, chance-corrected agreement statistics (*e.g.* Cohen's κ, Krippendorff's α) were not computed; full per-criterion agreement breakdowns are reported in Supplementary Note 4.

6 Real-world evaluation: Complex clinical cases

We assessed ATHENA-R1 on clinical case vignettes constructed at an academic medical center. Three physicians independently rated ATHENA-R1's responses on three cases selected for therapeutic complexity (competing physiological constraints, absent or incomplete clinical guidelines). Responses were scored on an absolute 1–5 Likert scale across eight criteria, without a comparison model.

6.1 Case construction

Case vignettes were constructed by a clinician at an academic medical center from the details of real adult and neonatal hospitalized patients. Each vignette was written in a structured clinical format (presenting problem, past medical history, medications, relevant laboratory and hemodynamic values, specific treatment task) and contained no direct identifiers, no dates, and no information meeting HIPAA's 18 identifier criteria. Vignettes were not verbatim chart extracts; they abstracted the clinical reasoning scenario from each source case while removing any potentially identifying detail.

From an initial set of five constructed vignettes, three were selected for this evaluation to represent distinct reasoning challenges: (1) selection of an ACE inhibitor in a post-CABG patient with recent contrast-induced nephropathy, CKD stage 2, and heart failure with reduced ejection fraction; (2) empirical antibiotic choice in a patient with a mechanical mitral valve on warfarin who developed a surgical-site infection after total knee arthroplasty, where levofloxacin's interaction with warfarin must be weighed against alternatives; and (3) β-blocker selection for secondary prevention in a post-STEMI patient with severe persistent asthma. Each case poses a treatment task that requires integrating patient-specific contraindications, drug-drug interactions, and competing guideline considerations. The full vignettes, reasoning traces, and final answers are reported in Supplementary Note 5.

6.2 Evaluation protocol

Three physicians at the same institution served as reviewers. Each reviewer received the three case vignettes together with ATHENA-R1's full response (final answer plus reasoning trace) and scored each response independently on eight criteria using a 1–5 Likert scale: task success, helpfulness of rationale, cognitive traceability, possibility of harm, alignment with clinical consensus, accuracy of content, completeness, and clinical relevance. The per-criterion definitions, scoring questions, and 1–5 anchors follow the rubric defined in Supplementary Note 4. No time limit was imposed, and reviewers were not blinded, as only ATHENA-R1's output was scored.

6.3 Ethical framework

The study used case vignettes constructed from clinical scenarios and did not involve contact with patients, access to identifiable patient data in the model inputs, or clinical intervention. The study was determined by the institutional review office to not constitute human subjects research.

6.4 Statistical analysis

For each of the eight criteria, we report the mean and standard deviation across three cases $\times$ three reviewers (9 ratings per criterion). Ratings that reviewers did not provide (2 of 72 cells) were excluded pairwise rather than imputed. Because this evaluation did not include a comparison model, no between-model statistical tests were performed; the study is presented as an absolute-rating assessment on three representative cases.

7 Real-world evaluation: Population-scale EHR analyses

We evaluated novel adverse event predictions generated by ATHENA-R1 by conducting retrospective cohort analyses on electronic health record (EHR) data, including diagnosis codes and medication purchase records, from Clalit Health Services.

7.1 Methodological plan

The analysis pipeline consisted of four stages: (1) selecting clinically relevant patient populations (defined by a primary disease, a comorbidity, and a drug); (2) using ATHENA-R1 to predict adverse events that these patient populations are at increased risk for; (3) defining these clinical concepts using standardized medical codes and constructing patient cohorts; and (4) performing statistical analyses to compare the risk of predicted adverse events between patient groups.

7.2 Design of patient subpopulations

To identify patient populations for which ATHENA-R1 could generate novel and testable hypotheses, a multi-step selection process was followed. First, the prevalence of all diseases in the Clalit EHR database was computed to identify common conditions with sufficient patient data. A clinician reviewed the most prevalent diseases and selected a subset for further analysis (*e.g.*, hypertension, diabetes mellitus). Next, a large language model was used to survey the medical literature and, for each selected disease, identify comorbidities known to influence disease subtypes or cause differential drug responses. These were manually reviewed to construct patient subpopulations defined by *(disease, comorbidity, drug)* triads. Note that "drug" could represent multiple drugs that are members of the same drug class.

7.3 Adverse event prediction with ATHENA-R1

For each selected patient triad, we used ATHENA-R1 to generate hypotheses of potential novel adverse events. For details on the generation, scoring, and ranking procedure for adverse event hypothesis generation, please see Supplementary Note 6.

7.4 Phenotype and cohort definitions

Phenotype mapping. All clinical entities (diseases, comorbidities, drugs, and predicted adverse events) were manually mapped to one or more corresponding medical codes. Diseases, comorbidities, and adverse events were defined using the International Classification of Diseases, Ninth Revision (ICD-9) and other internal coding systems (*e.g.*, ICPC, CHR). Drugs were mapped to their Anatomical Therapeutic Chemical (ATC) Level 5 codes. Notably, all entities were manually curated; for example,

drugs not prescribed in the Clalit system were excluded, and internationally prescribed medications relevant to the analysis (*e.g.*, cilazapril, oxprenolol, vildagliptin, cerivastatin) were included.

Cohort construction. For each *(disease, comorbidity, drug)* triad, we first constructed a base disease cohort containing all patients with at least one diagnosis code for the specified disease. Within this cohort, we then identified the dates of the following events for each patient, if they occurred:

- **Base disease diagnosis.** Date of the first recorded diagnosis for the primary disease.
- **Comorbidity diagnosis.** Date of the first recorded diagnosis for the comorbidity.
- **Drug exposure.** Date of the first recorded medication purchase for any of the specified drugs.

No temporal ordering was enforced between the diagnosis of the primary disease, the comorbidity, or the start of drug treatment.

7.5 Statistical analyses

Adverse event prevalence. We then calculated adverse event prevalence within five progressively specific cohorts:

- **General population.** We first calculated the overall lifetime population prevalence of each adverse event across all $5,437,870$ patients in the Clalit database.
- **Disease (D_1).** Patients with the base disease. An adverse event was counted if its first diagnosis occurred after the patient's first diagnosis of the primary disease.
- **Disease + drug ($D_1 + R$).** Patients with both the base disease and exposure to the drug. An adverse event was counted if its first diagnosis occurred after the latest of the primary disease diagnosis or first drug purchase dates.
- **Disease + comorbidity ($D_1 + D_2$).** Patients with both the base disease and the comorbidity. An adverse event was counted if its first diagnosis occurred after the latest of the primary disease or comorbidity diagnosis dates.

- **Disease + comorbidity + drug ($D_1 + D_2 + R$).** Patients with the base disease, the comorbidity, and exposure to the drug. An adverse event was counted if its first diagnosis occurred after the latest of the primary disease diagnosis, comorbidity diagnosis, or first drug purchase dates.

Prevalence was calculated as the number of patients in a cohort who experienced the adverse event divided by the total number of patients in that cohort, expressed as a percentage. Results were visualized using bar plots comparing prevalence across the five groups.

Regression analysis. To assess the association between drug exposure and the risk of an adverse event, we compared:

- **Exposed group.** Patients with the disease, comorbidity, and drug exposure. The index date for this group was defined as the latest of the primary disease diagnosis, comorbidity diagnosis, or first drug purchase dates.
- **Unexposed group.** Patients with the disease and comorbidity, but no record of purchasing the specified drug(s). The index date for this group was the latest of the primary disease or comorbidity diagnosis dates.

To account for potential confounding factors, we fit a multivariable logistic regression model for each adverse event, run using the `statsmodels` library in Python. By default, all regression models were adjusted by various confounding factors:

- **Age.** Calculated as a continuous variable at the patient's index date.
- **Sex.** Included as a categorical variable.
- **Socioeconomic status (SES).** This variable was pre-processed by first categorizing it into tertiles (low, intermediate, high) based on the distribution within the base disease population. Any missing SES values were imputed to the "intermediate" category. SES was then included in the model as a categorical variable.
- **Healthcare utilization.** Calculated as a continuous variable as the number of outpatient visits per year, excluding emergency room encounters and with de-duplication by day, such that multiple events on the same calendar date are accounted for as one visit.

The logistic regression model was fit separately for each adverse event to estimate the adjusted odds ratio and 95% confidence interval for the association between drug exposure and the outcome, visualized using forest plots.

References

1. Brown, T. *et al.* Language models are few-shot learners. *Advances in Neural Information Processing Systems* **33,** 1877–1901 (2020).
2. Kass-Hout, T. A. *et al.* OpenFDA: an innovative platform providing access to a wealth of FDA's publicly available data. *Journal of the American Medical Informatics Association* **23,** 596–600 (2016).
3. Ochoa, D. *et al.* The next-generation Open Targets Platform: reimagined, redesigned, rebuilt. *Nucleic Acids Research* **51,** D1353–D1359. doi:10.1093/nar/gkac1046 (2023).
4. Gargano, M. A. *et al.* The Human Phenotype Ontology in 2024: phenotypes around the world. *Nucleic Acids Research* **52,** D1333–D1346. doi:10.1093/nar/gkad1005 (2024).
5. Chandak, P., Huang, K. & Zitnik, M. Building a knowledge graph to enable precision medicine. *Scientific Data.* doi:https://doi.org/10.1038/s41597-023-01960-3 (2023).
6. U.S. Food and Drug Administration. *openFDA* 2024.
7. Targets, O. *24.09 Platform Release Now Live* https://community.opentargets.org/t/24-09-platform-release-now-live/1556?ref=blog.opentargets.org. 2024.
8. Li, Z., Zhang, X., Zhang, Y., Long, D., Xie, P. & Zhang, M. Towards general text embeddings with multi-stage contrastive learning. *arXiv preprint arXiv:2308.03281* (2023).
9. Vaswani, A. *et al. Attention is All you Need* in *Advances in Neural Information Processing Systems* **30** (Curran Associates, Inc., 2017).
10. Hu, E. J. *et al. LoRA: Low-Rank Adaptation of Large Language Models* in *International Conference on Learning Representations (ICLR)* (2022).
11. Von Werra, L. *et al. TRL: Transformer Reinforcement Learning* https://github.com/huggingface/trl. 2020.
12. Tunstall, L. *et al. The Alignment Handbook* version 0.3.0.dev0. 2023.
13. Wolf, T. *et al. Transformers: State-of-the-Art Natural Language Processing* in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations* (Association for Computational Linguistics, Online, 2020), 38–45.
14. Rasley, J., Rajbhandari, S., Ruwase, O. & He, Y. *Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters* in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2020), 3505–3506.
15. Paszke, A. *et al. Automatic differentiation in PyTorch* in *NIPS-W* (2017).
16. Guo, D. *et al.* DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature* **645,** 633–638. doi:10.1038/s41586-025-09422-z (2025).
17. Zhao, Y. *et al. Geometric-mean policy optimization* in *International Conference on Learning Representations (ICLR)* (2026).

18. Sheng, G. *et al. HybridFlow: A Flexible and Efficient RLHF Framework* in *Proceedings of the Twentieth European Conference on Computer Systems* (2025), 1279–1297. doi:10.1145/3689031.3696075.