

# Machine Programming

Lecture 20 – Review & Special Topics on Synthesizing Diverse DSLs

# Recap & Review

# Foundations + Applications

of machine programming

# Topics

- Foundations

- **Programming language, syntax and semantics**
- **Classical synthesis**, e.g., inductive synthesis, bottom-up and top-down synthesis, type-guided synthesis, specification guided synthesis
- **LLM-based synthesis**, e.g., natural language guided synthesis, agentic frameworks, model context protocol, language server protocol

- Applications

- **Software Engineering & Security**, e.g., testing, transpilation, verification
- **Math & Theorem Proving**, e.g., auto-formalization, proof search
- **Planning & Cyber-Physical Systems**, e.g. robotics, simulation, reward
- **Long-horizon Coding Tasks, Research**, and etc.



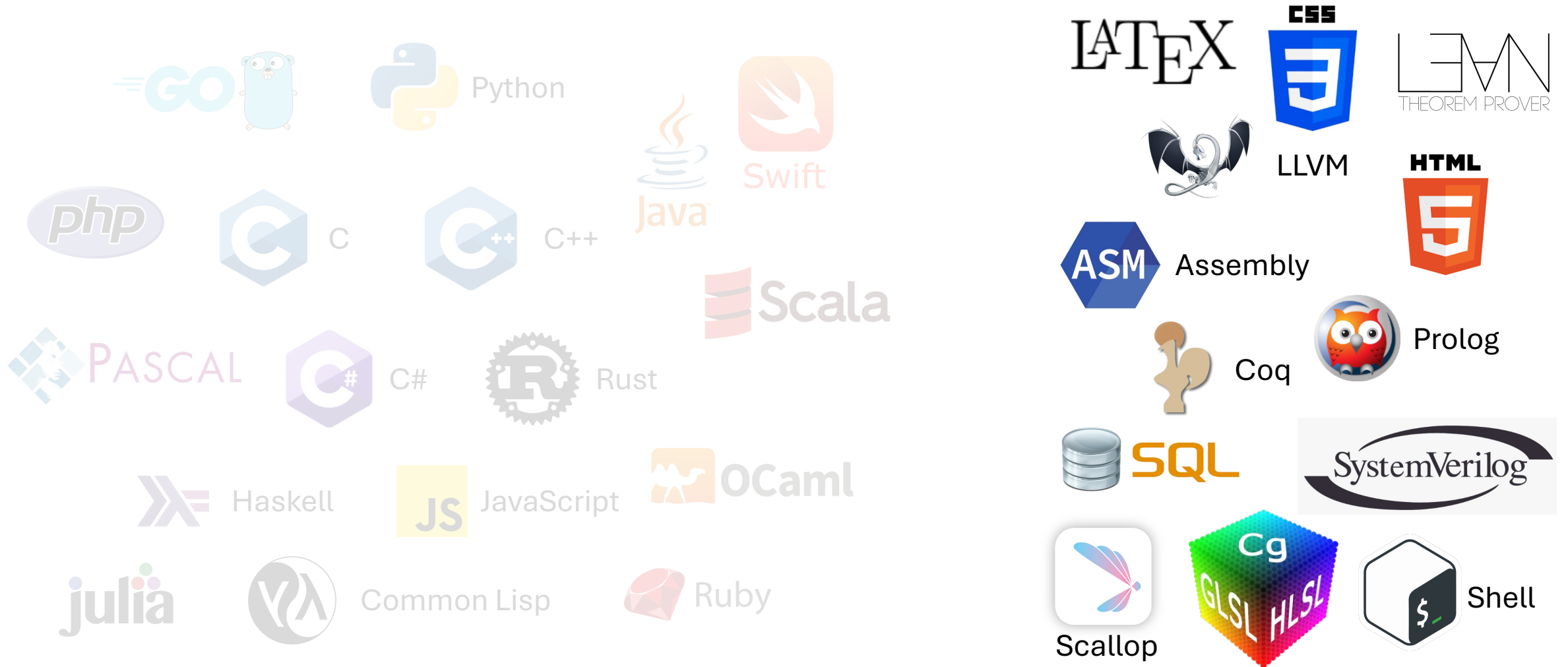
# Programming Languages



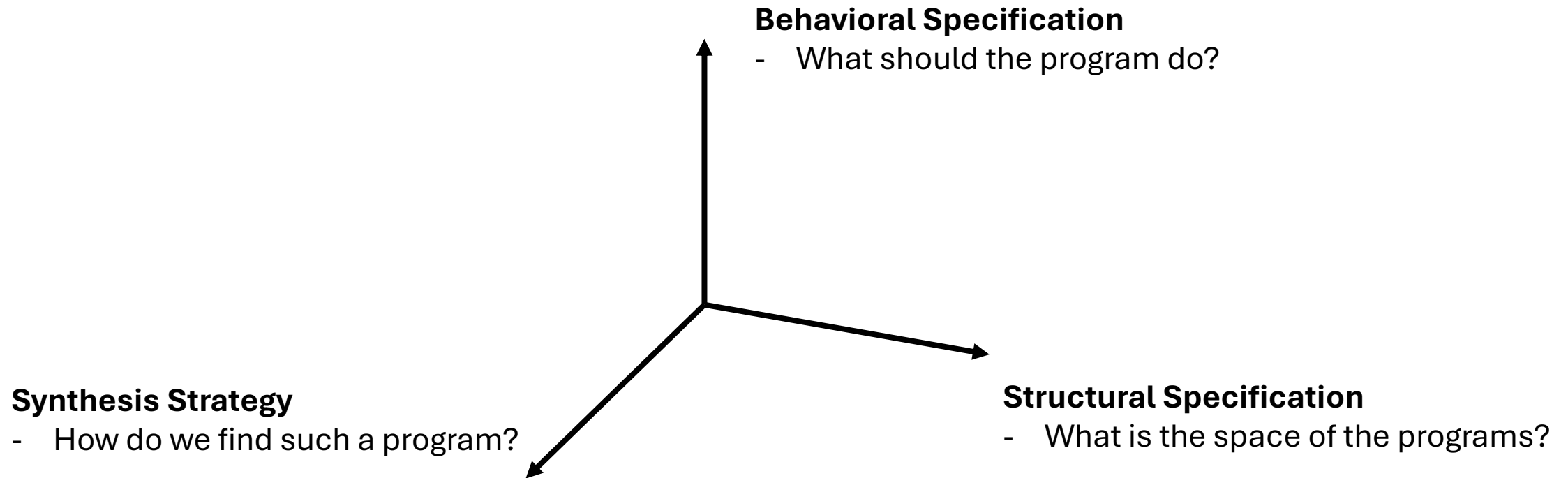
# General Purpose Programming Languages



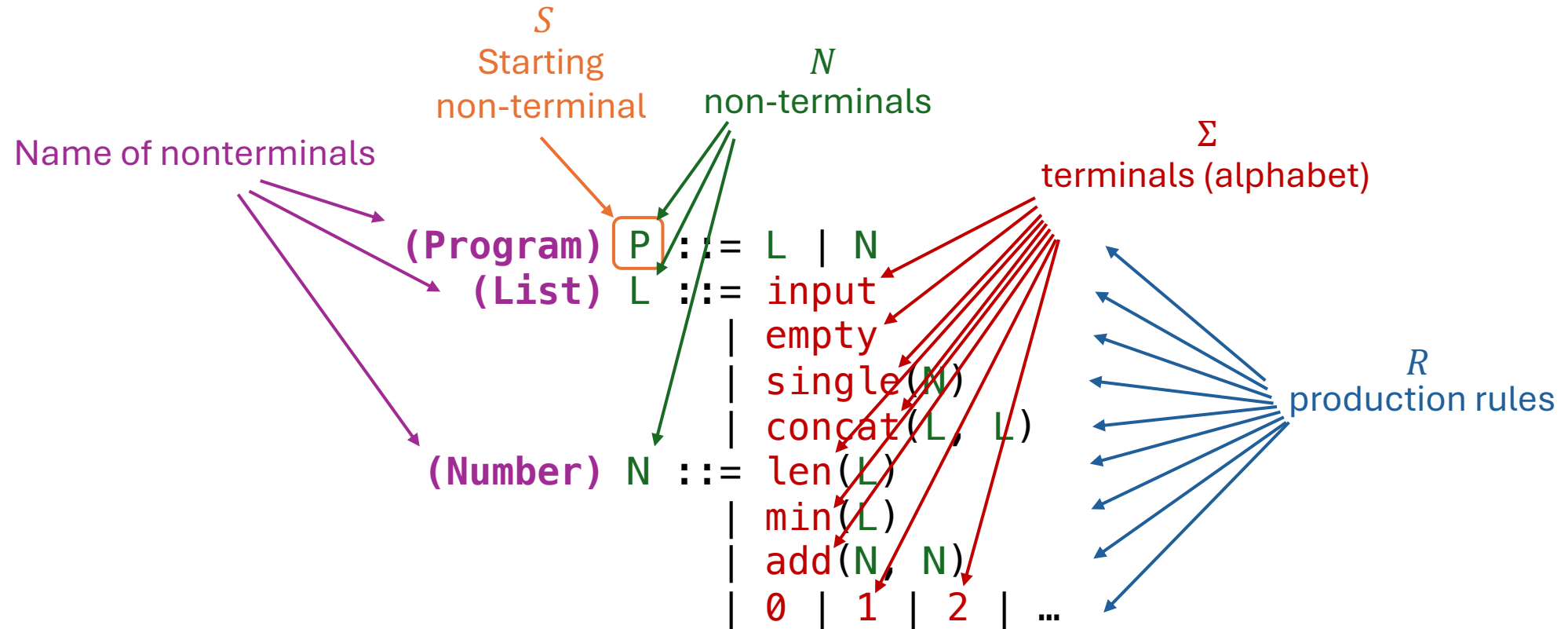
# Domain Specific Programming Languages



# Dimensions in Program Synthesis



# Syntax: Regular tree grammars (RTGs)



# Semantics: Meaning of a Language

(Program)  $\underline{P} ::= L \mid N$   
(List)  $L ::= \text{input}$   
           $\mid \text{empty}$   
           $\mid \text{single}(N)$   
           $\mid \text{concat}(L, L)$   
(Number)  $N ::= \text{len}(L)$   
           $\mid \text{min}(L)$   
           $\mid \text{add}(N, N)$   
           $\mid 0 \mid 1 \mid 2 \mid \dots$

$[[\cdot]] : T_{\Sigma} \times \text{IntList} \rightarrow \text{List} \mid \text{Int}$   
 $[[\text{input}]](x) = x$   
 $[[\text{empty}]](x) = []$   
 $\forall \tau \in T_{\Sigma}, [[\text{single}(\tau)]](x) = [ [[\tau]](x) ]$   
 $\forall \tau_1, \tau_2 \in T_{\Sigma}, [[\text{concat}(\tau_1, \tau_2)]](x) = [[\tau_1]](x) + [[\tau_2]](x)$   
 $\forall \tau \in T_{\Sigma}, [[\text{len}(\tau)]](x) = | [[\tau]](x) |$   
 $\forall \tau \in T_{\Sigma}, [[\text{min}(\tau)]](x) = \min_v (v \in [[\tau]](x))$   
 $\forall \tau_1, \tau_2 \in T_{\Sigma}, [[\text{add}(\tau_1, \tau_2)]](x) = [[\tau_1]](x) + [[\tau_2]](x)$   
 $[[0]](x) = 0, \dots$

Denotational Semantics

Mathematical meaning to each program construct

# Bottom-up enumeration: algorithm

```
bottom-up(< $\Sigma$ , N, R, S>, [ $i \rightarrow o$ ], max_depth):  
    bank := {}  
    for depth in [0..max_depth]:  
        forall rule in R:  
            forall new_prog in grow(rule, depth, bank):  
                if ( $A = S \wedge$  new_prog( $i$ ) =  $o$ ):  
                    return new_program  
            insert new_program to bank;  
  
grow( $A \rightarrow \sigma(A_1 \dots A_k)$ , d, bank):  
    if ( $d = 0 \wedge k = 0$ ) yield  $\sigma$  // terminal  
    else forall  $\langle t_1, \dots, t_k \rangle$  in bankk: // cartesian product  
        if  $A_i \rightarrow^* t_i$ :  
            yield  $\sigma(t_1, \dots, t_k)$ 
```

```
(Program)  $\underline{P} ::= L \mid N$   
      (List)  $L ::=$  input  
              | empty  
              | single( $N$ )  
              | concat( $L, L$ )  
      (Number)  $N ::=$  len( $L$ )  
                  | min( $L$ )  
                  | add( $N, N$ )  
                  | 0 | 1 | 2 | ...
```

# Type-Guided Top-Down Search

```
Expr ::= Var
      | lambda Var: Expr
      | filter(Expr, Expr)
      | map(Expr, Expr)
      | reduce(Expr, Expr, Expr)
      | Expr if Expr else Expr
      | Expr BinaryOp Expr
      | true | false
      | 0 | 1 | 2 | ...
```

**dropmin : List[List[Int]] -> List[List[Int]]**

dropmin(list) = <Expr>

[ [71, 75, 83], [75, 83],  
[90, 87, 95], → [90, 95],  
[68, 77, 80] ] [77, 80] ]

list  
✗

lambda <Var>: <Expr>  
**List[List[Int]]**

filter(<Expr>, <Expr>)

<Expr> if <Expr> else <Expr> ...

$$[\text{Lambda}] \frac{C \cup \{x \mapsto \tau_1\} \vdash e : \tau_2}{C \vdash \lambda x. e : \tau_1 \rightarrow \tau_2}$$



# Verification of Program with a Specification

{Pre-condition} Program {Post-condition}

$$\{P\} c \{Q\}$$

Program  $c$ : a program state transformer

Pre- and Post-condition  $P, Q$ : a Boolean function over program states

$$\forall x, P(x) \Rightarrow Q(c(x))$$

Program  $c$ : a program state transformer  
Pre- and Post-condition  $P, Q$ : a Boolean function over program states

$$\forall x, P(x) \Rightarrow Q(c(x))$$

$P$

$x$ : Int

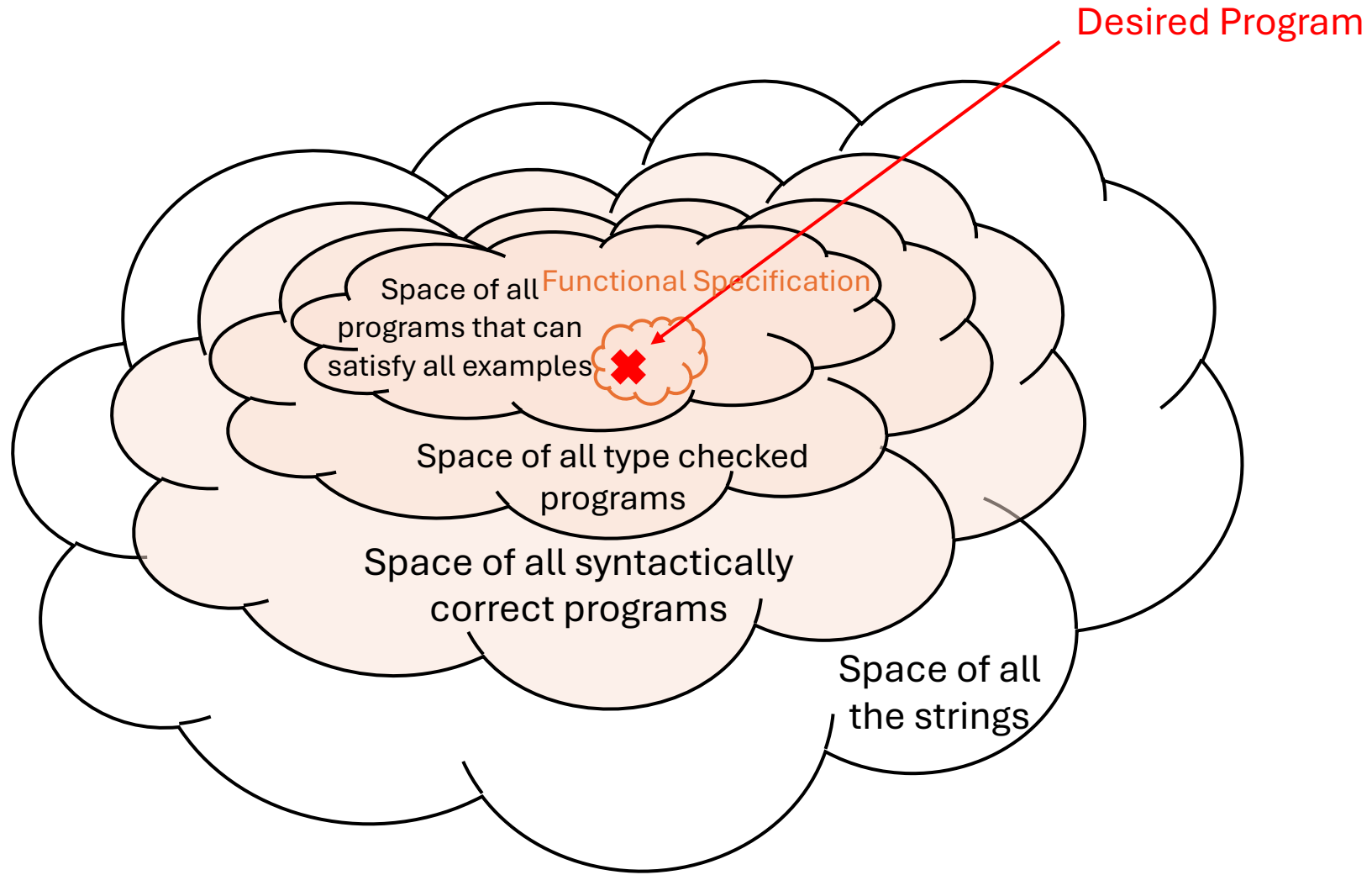
```
int abs(int x) {  
    int y;  
    if (x >= 0) // B1  
        y = x; // B2  
    else  
        y = -x; // B3  
    return y; // B4  
}
```

$c$

$Q$

$(y = -x \vee y = x) \wedge y \geq 0$

# High Level Picture



# Modeling Language

Problem Definition: **Next Token Prediction**

Input: **Prefix**  $t_1, t_2, \dots, t_{i-1}$

Output: **Next Token**  $t_i$

Goal: Compute the probability  $\Pr(t_i \mid t_1, t_2, \dots, t_{i-1})$



# TRANSFORMER EXPLAINER

Examples ▾

for i in range(10): print(i)

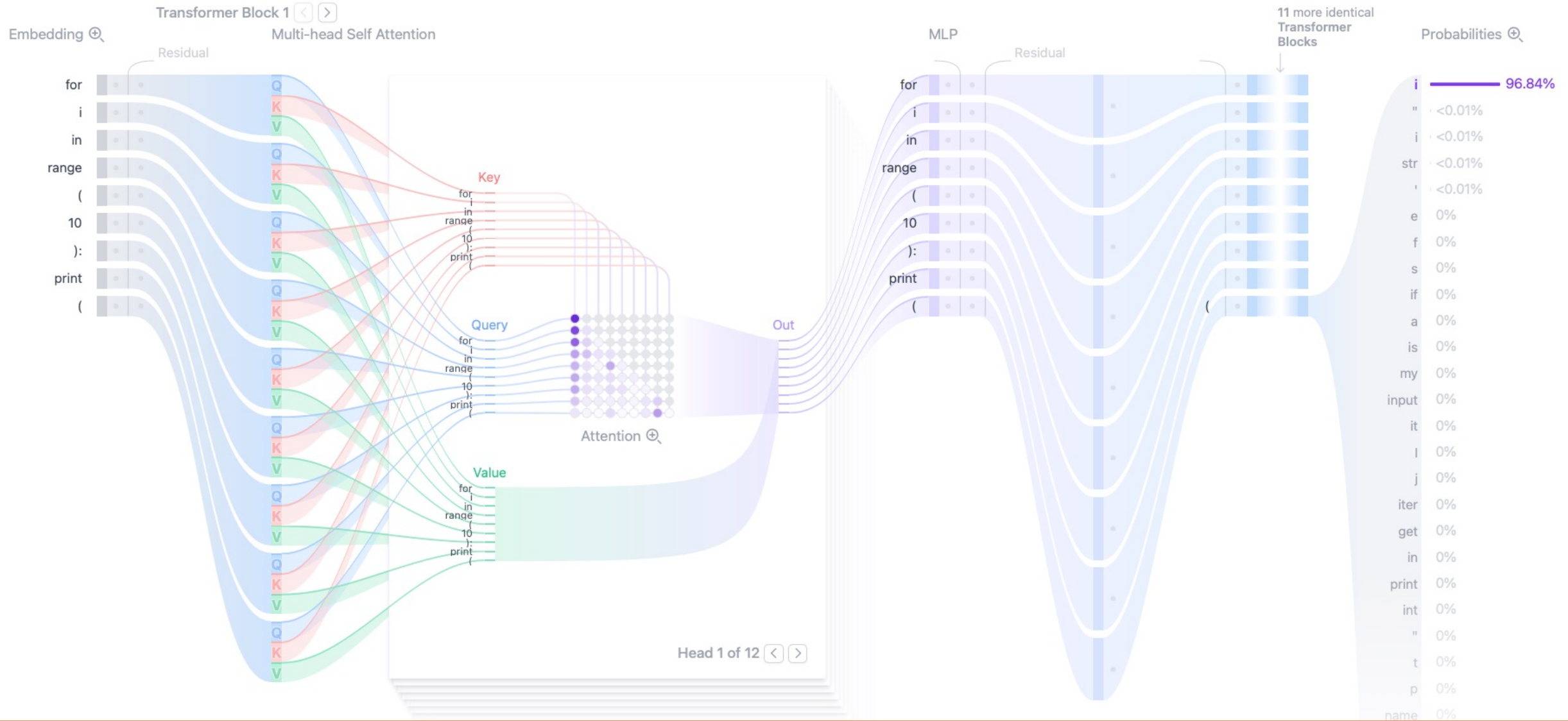
Generate

Temperature

0.8

Sampling ☒ Top-k ☐ Top-p

k=5



# Synthesis with a Conversation

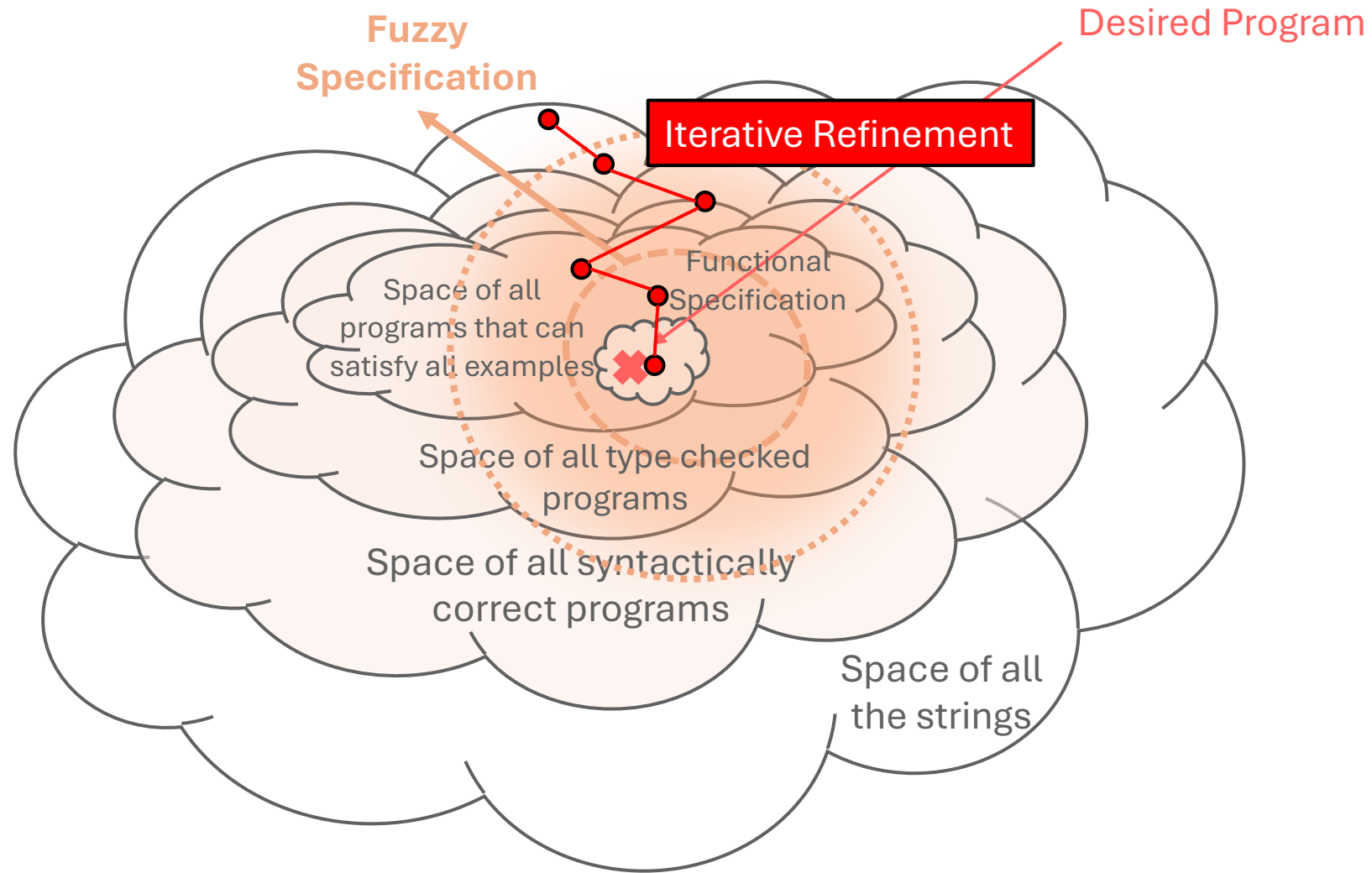
## Special Tokens denoting Roles

```
{
  "messages": [
    { "role": "system",
      "parts": ["You are a senior software engineer from..."] },
    { "role": "user",
      "parts": ["Write this function:\ndef add(a, b):"] },
    { "role": "assistant",
      "parts": ["def add(a, b): return a + b"] },
    { "role": "user",
      "parts": ["Write this function:\ndef reverse(s: str)..."] },
    { "role": "assistant",
      "parts": ["def reverse(s: str) -> str:\n output = \"\"\n for c in s:\n output = c + output\n return output"] },
    { "role": "user",
      "parts": ["Write this function:\ndef is_palindrome(s: str)"] }
  ]
}
```

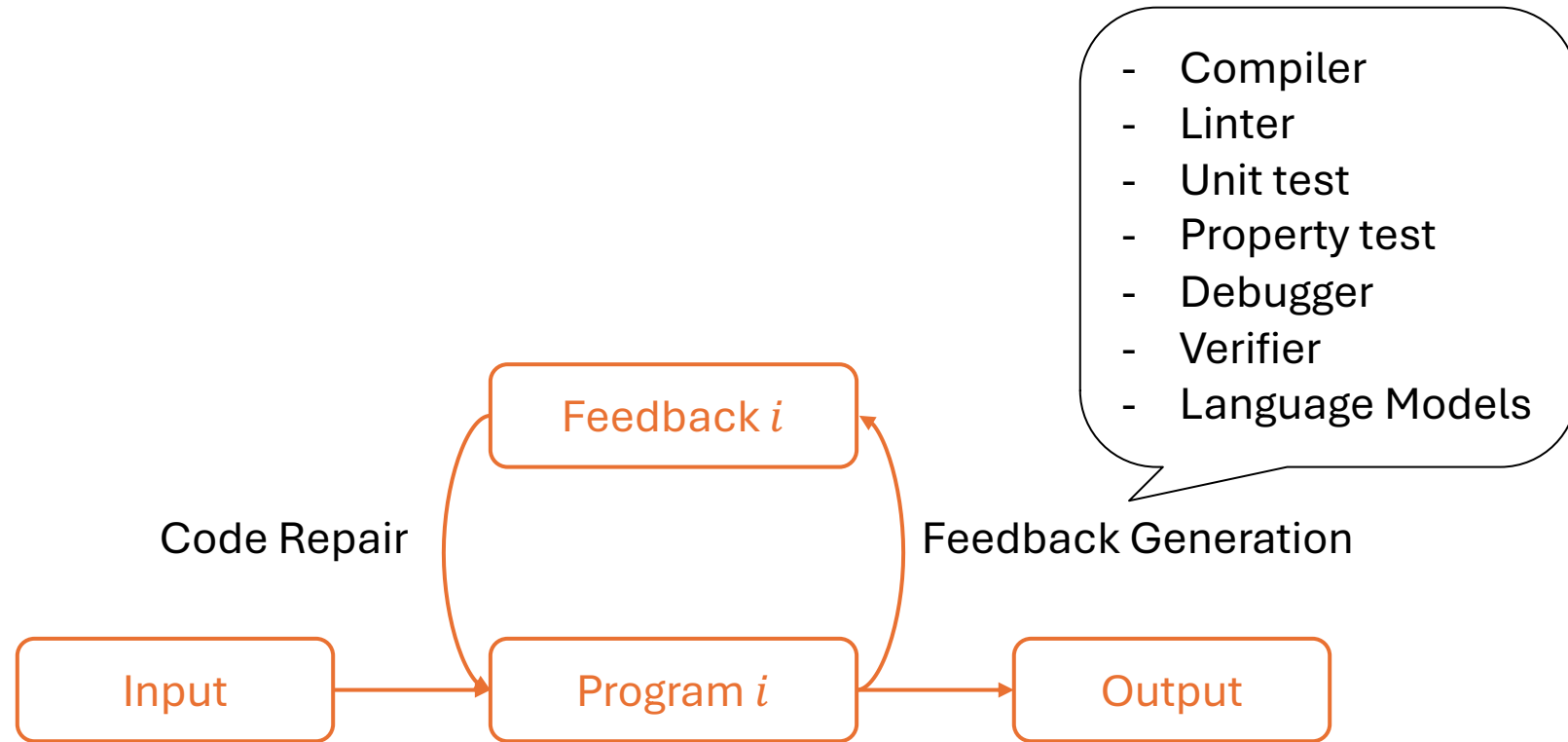


```
<system>
You are a senior software engineer from...
<user>
Write this function:
  def add(a, b):
<assistant>
def add(a, b): return a + b
<user>
Write this function:
  def reverse(s: str)...
<assistant>
def reverse(s: str) -> str:
  output = ""
  for c in s:
    output = c + output
  return output
<user>
Write this function:
  def is_palindrome(s: str)
```

# High Level Picture



# Feedback for Iterative Refinement





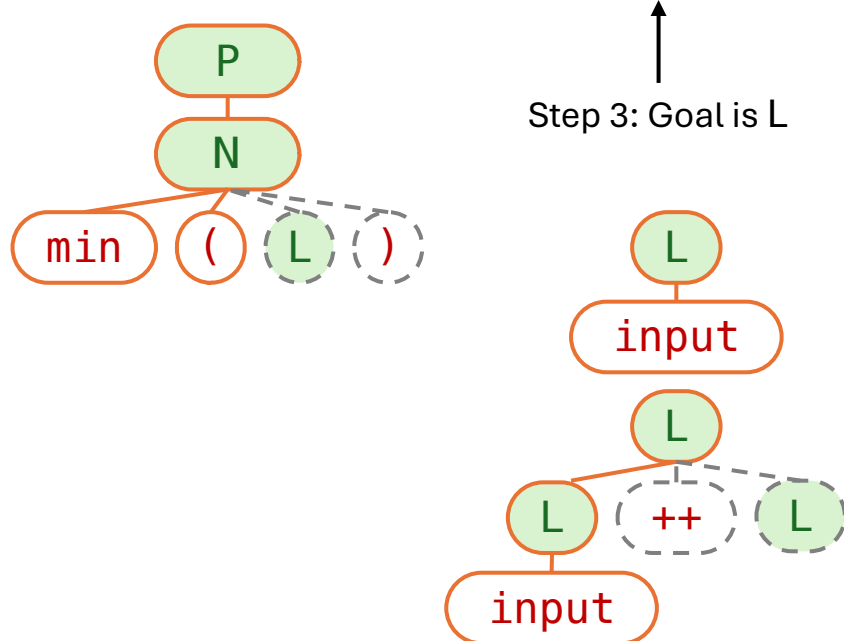
# Constrained Decoding

Partial Program

`min(input ++ [0])`

Token Sequence

`['min', '(', 'input', '++', '[', '0', ']']`



```
(Program) P <- L | N
(List) L <- 'input'
          | '[' ']'
          | '[' N ']'
          | L '++' L
(Number) N <- 'len' '(' L ')'
            | 'min' '(' L ')'
            | N '+' N
            | '0' | '1' | '2' | ...
```

We can expand **L** into **input**, matching the third token

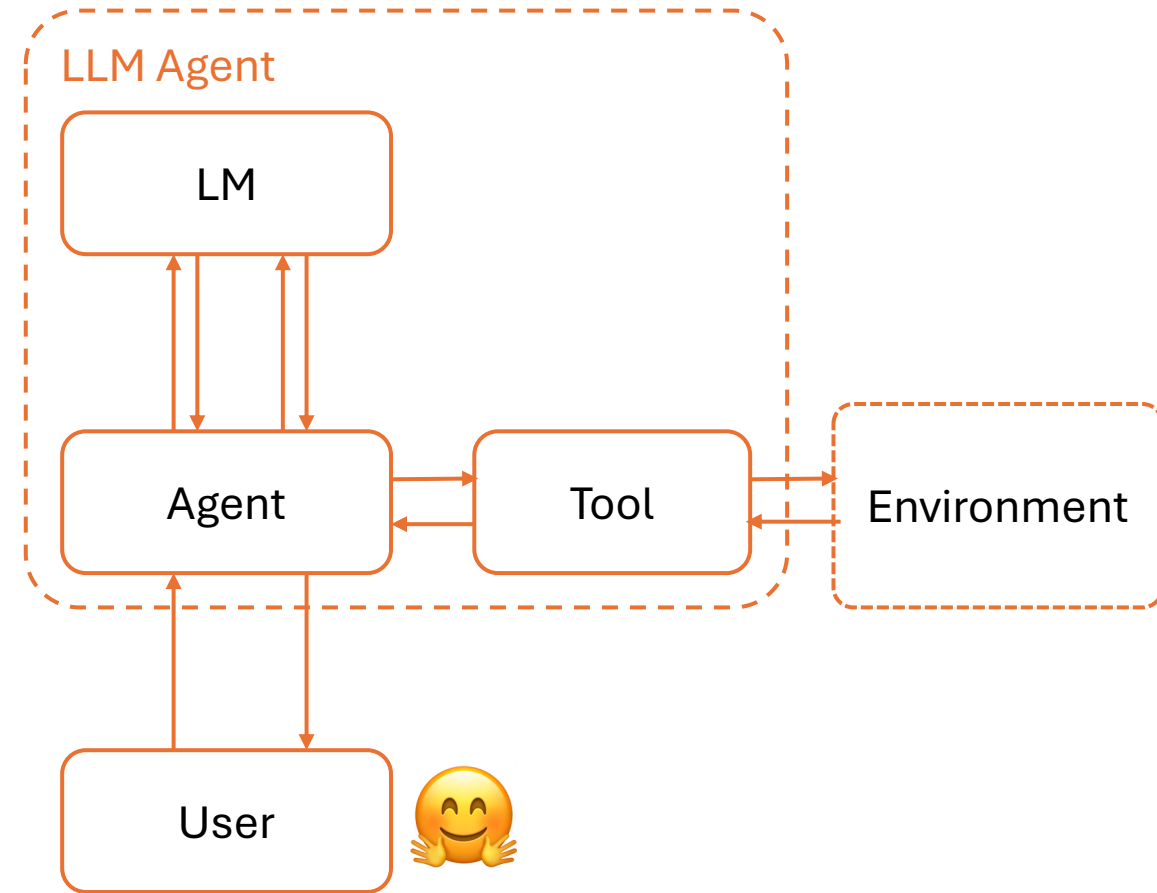
➔ Finishing L, next token is `)`

We can expand **L** into **L + L**, where the first **L** can be expanded into **input**, matching the third token

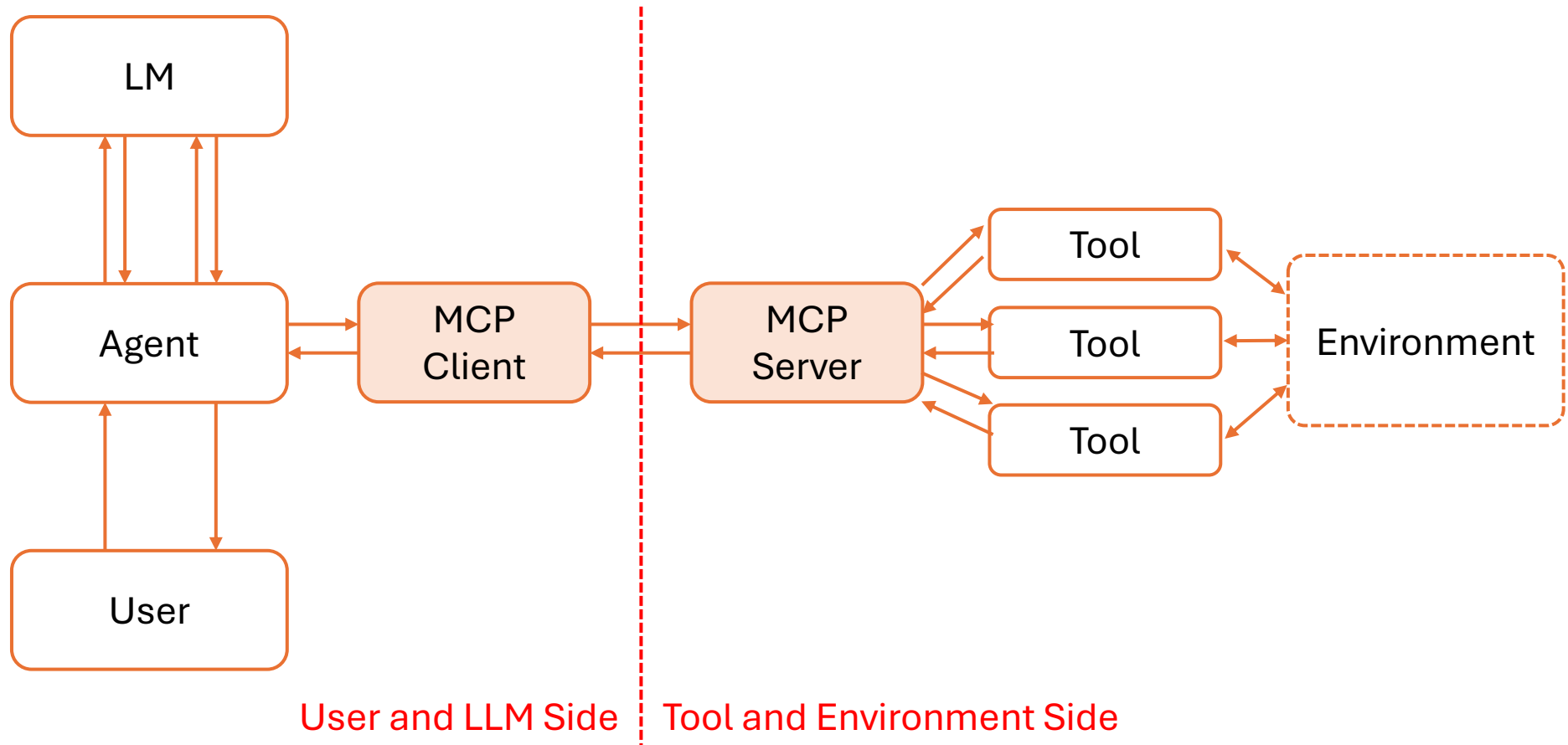
➔ Finishing the left L, next token is `++`

# LLM-based Agentic Frameworks

- **User** (human, end-user)
- **Agent** (a computer program)
- **LM** (a neural language model)
- **Environment** (the world)
- **Tool** (access point to the world)



# Model Context Protocol (MCP)



## Third-Party Servers

### Official Integrations

Official integrations are maintained by companies building production ready MCP servers for their platforms.

- 🔗 **21st.dev Magic** - Create crafted UI components inspired by the best 21st.dev design engineers.
- 🔗 **ActionKit by Paragon** - Connect to 130+ SaaS integrations (e.g. Slack, Salesforce, Gmail) with Paragon's **ActionKit** API.
- 🔗 **Adfin** - The only platform you need to get paid - all payments in one place, invoicing and accounting reconciliations with **Adfin**.
- 🔗 **AgentOps** - Provide observability and tracing for debugging AI agents with **AgentOps** API.
- 🔗 **AgentQL** - Enable AI agents to get structured data from unstructured web with **AgentQL**.
- 🔗 **AgentRPC** - Connect to any function, any language, across network boundaries using **AgentRPC**.
- 🔗 **Agentset** - RAG for your knowledge base connected to **Agentset**.
- 🔗 **Aiven** - Navigate your **Aiven projects** and interact with the PostgreSQL®, Apache Kafka®, ClickHouse® and OpenSearch® services
- 🔗 **Alation** - Unlock the power of the enterprise Data Catalog by harnessing tools provided by the Alation MCP server.
- 🔗 **Alby Bitcoin Payments** - Connect any bitcoin lightning wallet to your agent to send and receive instant payments globally with your agent.
- 🔗 **Algolia** - Use AI agents to provision, configure, and query your **Algolia** search indices.
- 🔗 **Alibaba Cloud AnalyticDB for MySQL** - Connect to an **AnalyticDB for MySQL** cluster for getting database or table metadata, querying and analyzing data. It will be supported to add the OpenAPI for cluster operation in the future.
- 🔗 **Alibaba Cloud AnalyticDB for PostgreSQL** - An MCP server to connect to **AnalyticDB for PostgreSQL** instances, query and analyze data.
- 🔗 **Alibaba Cloud DataWorks** - A Model Context Protocol (MCP) server that provides tools for AI, allowing it to interact with the **DataWorks** Open API through a standardized interface. This implementation is based on the Alibaba Cloud Open API and enables AI agents to perform cloud resources operations seamlessly.
- 🔗 **Alibaba Cloud OpenSearch** - This MCP server equips AI Agents with tools to interact with **OpenSearch** through a standardized and extensible interface.
- 🔗 **Alibaba Cloud OPS** - Manage the lifecycle of your Alibaba Cloud resources with **CloudOps Orchestration Service** and Alibaba Cloud OpenAPI.
- 🔗 **Alibaba Cloud RDS** - An MCP server designed to interact with the Alibaba Cloud RDS OpenAPI, enabling programmatic management of RDS resources via an LLM.
- 🔗 **AlipayPlus** - Connect your AI Agents to AlipayPlus Checkout Payment.
- 🔗 **AlVoiceLab** - An AI voice toolkit with TTS, voice cloning, and video translation, now available as an MCP server for smarter agent integration.
- 🔗 **Alpaca** - Alpaca's MCP server lets you trade stocks and options, analyze market data, and build strategies through **Alpaca's Trading API**
- 🔗 **AlphaVantage** - Connect to 100+ APIs for financial market data, including stock prices, fundamentals, and more from **AlphaVantage**
- 🔗 **AltTester®** - Use AltTester® capabilities to connect and test your Unity or Unreal game. Write game test automation faster and smarter, using **AltTester** and the AltTester® MCP server.
- 🔗 **Antom** - Connect your AI Agents to Antom Checkout Payment.
- 🔗 **Anypoint** - An MCP server enabling AI assistants to interact with **Anypoint** - a local and collaborative wiki - to organize objects, lists, and more through natural language.
- 🔗 **Apache Doris** - MCP Server For **Apache Doris**, an MPP-based real-time data warehouse.
- 🔗 **Apache IoTDB** - MCP Server for **Apache IoTDB** database and its tools
- 🔗 **Apache Pinot** - MCP server for running real - time analytics queries on Apache Pinot, an open-source OLAP database built for high-throughput, low-latency powering real-time applications.
- 🔗 **Apify** - Use 6,000+ pre-built cloud tools to extract data from websites, e-commerce, social media, search engines, maps, and more
- 🔗 **APIMatic MCP** - APIMatic MCP Server is used to validate OpenAPI specifications using **APIMatic**. The server processes OpenAPI files and returns validation summaries by leveraging APIMatic's API.
- 🔗 **Apollo MCP Server** - Connect your GraphQL APIs to AI agents
- 🔗 **Aqara MCP Server** - Control **Aqara** smart home devices, query status, execute scenes, and much more using natural language.
- 🔗 **Archbee** - Write and publish documentation that becomes the trusted source for instant answers with AI. Stop cobbling tools and use **Archbee** — the first complete

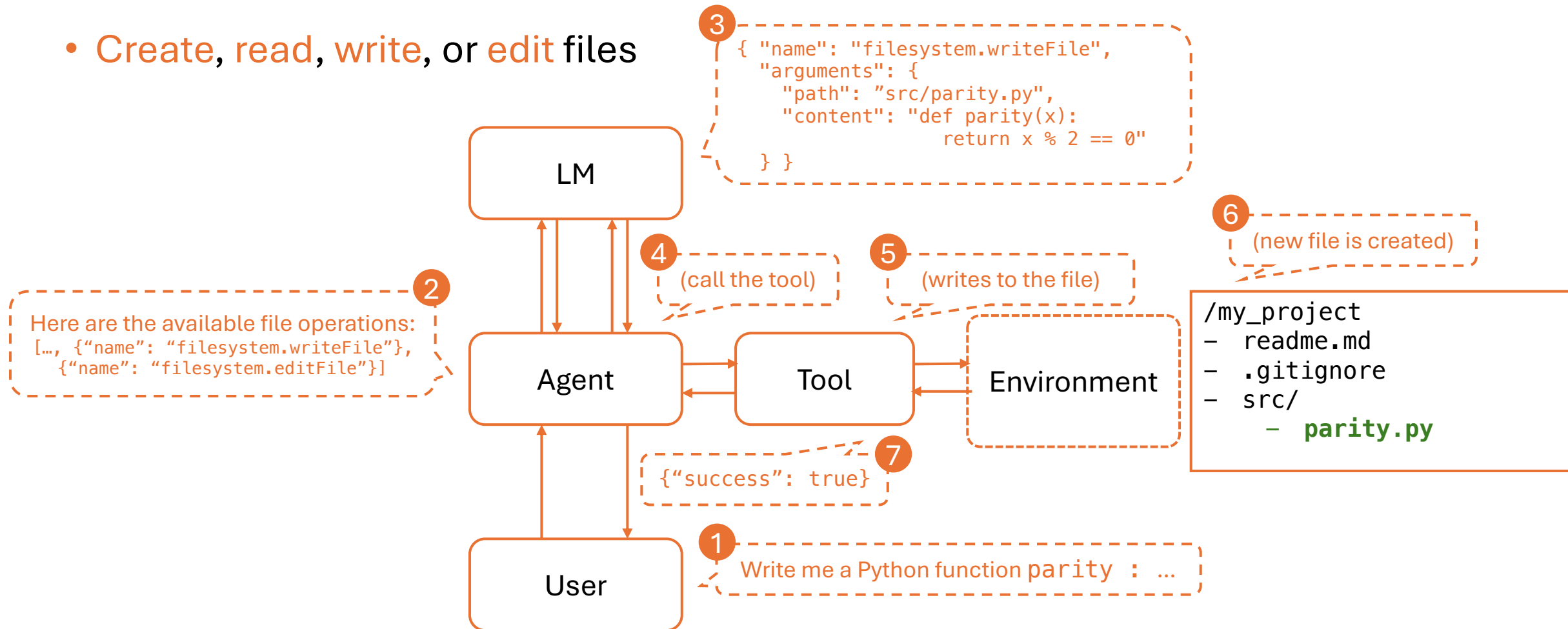
- 🔗 **Astra DB** - Comprehensive tools for managing collections and documents in a **AstraDB** NoSQL database with a full range of operations such as create, update, delete, find, and associated bulk actions.
- 🔗 **Atla** - Enable AI agents to interact with the **Atla API** for state-of-the-art LLMJ evaluation.
- 🔗 **Atlan** - The Atlan Model Context Protocol server allows you to interact with the **Atlan** servers through multiple tools.
- 🔗 **Atlassian** - Securely interact with Jira work items and Confluence pages, and search across both.
- 🔗 **AtomGit** - Official AtomGit server for integration with repository management, PRs, issues, branches, labels, and more.
- 🔗 **Audiense Insights** - Marketing insights and audience analysis from **Audiense** reports, covering demographic, cultural, influencer, and content engagement analysis.
- 🔗 **Auth0** - MCP server for interacting with your Auth0 tenant, supporting creating and modifying actions, applications, forms, logs, resource servers, and more.
- 🔗 **Authenticator App - 2FA** - A secure MCP (Model Context Protocol) server that enables AI agents to interact with the Authenticator App.
- 🔗 **AWS** - Specialized MCP servers that bring AWS best practices directly to your development workflow.
- 🔗 **Axiom** - Query and analyze your Axiom logs, traces, and all other event data in natural language
- 🔗 **Azure** - The Azure MCP Server gives MCP Clients access to key Azure services and tools like Azure Storage, Cosmos DB, the Azure CLI, and more.
- 🔗 **Azure DevOps** - Interact with Azure DevOps services like repositories, work items, builds, releases, test plans, and code search.
- 🔗 **Backdocket** - Search, Retrieve, and Update your **Backdocket** data. This currently includes Claims, Matters, Contacts, Tasks and Advanced Searches. To easily use the Remote Mcp Server utilize the following url: **https://ai.backdocket.com/mcp**
- 🔗 **Baidu Map - Baidu Map MCP Server** provides tools for AI agents to interact with Baidu Maps APIs, enabling location-based services and geospatial data analysis.
- 🔗 **Bankless Onchain** - Query Onchain data, like ERC20 tokens, transaction history, smart contract state.
- 🔗 **Baserow** - Query data from Baserow self-hosted or SaaS databases using MCP integration.
- 🔗 **BICScan** - Risk score / asset holdings of EVM blockchain address (EOA, CA, ENS) and even domain names.
- 🔗 **Bitrise** - Chat with your builds, CI, and **more**.
- 🔗 **Boikot** - Learn about the ethical and unethical actions of major companies with **boikot.xyz**.
- 🔗 **BoldSign** - Search, request, and manage e-signature contracts effortlessly with **BoldSign**.
- 🔗 **Boost.space** - An MCP server integrating with **Boost.space** for centralized, automated business data from 2000+ sources.
- 🔗 **Box** - Interact with the Intelligent Content Management platform through Box AI.
- 🔗 **BrightData** - Discover, extract, and interact with the web - one interface powering automated access across the public internet.
- 🔗 **Browserbase** - Automate browser interactions in the cloud (e.g. web navigation, data extraction, form filling, and more)
- 🔗 **BrowserStack** - Access BrowserStack's **Test Platform** to debug, write and fix tests, do accessibility testing and more.
- 🔗 **Buildkite** - Exposing Buildkite data (pipelines, builds, jobs, tests) to AI tooling and editors.
- 🔗 **Buildable** (TypeScript) - Official MCP server for Buildable AI-powered development platform. Enables AI assistants to manage tasks, track progress, get project context, and collaborate with humans on software projects.
- 🔗 **BuiltWith** - Identify the technology stack behind any website.
- 🔗 **Burp Suite** - MCP Server extension allowing AI clients to connect to **Burp Suite**
- 🔗 **Cal.com** - Connect to the Cal.com API to schedule and manage bookings and appointments.
- 🔗 **Campertunity** - Search campgrounds around the world on campertunity, check availability, and provide booking links.
- 🔗 **Canva** - Provide AI - powered development assistance for **Canva** apps and integrations.
- 🔗 **Carbon Voice** - MCP Server that connects AI Agents to **Carbon Voice**. Create, manage, and interact with voice messages, conversations, direct messages, folders, voice memos, AI actions and more in **Carbon Voice**.
- 🔗 **Cartesia** - Connect to the **Cartesia** voice platform to perform text-to-speech, voice cloning etc.
- 🔗 **Cashfree** - **Cashfree Payments** official MCP server.
- 🔗 **CB Insights** - Use the **CB Insights** MCP Server to connect to **ChatGPT**

- 🔗 **Claudia** - Exposes **Claudia**'s media output, transformation, AI analysis, management, optimization and delivery as tools usable by AI agents
- 🔗 **Cloudsway SmartSearch** - Web search MCP server powered by Cloudsway, supporting keyword search, language, and safety options. Returns structured JSON results.
- 🔗 **Codacy** - Interact with **Codacy** API to query code quality issues, vulnerabilities, and coverage insights about your code.
- 🔗 **CodeLogic** - Interact with **CodeLogic**, a Software Intelligence platform that graphs complex code and data architecture dependencies, to boost AI accuracy and insight.
- 🔗 **CoinGecko** - Official **CoinGecko API** MCP Server for Crypto Price & Market Data, across 200+ Blockchain Networks and 8M+ Tokens.
- 🔗 **Comet Opik** - Query and analyze your **Opik** logs, traces, prompts and all other telemetry data from your LLMs in natural language.
- 🔗 **Conductor** - Interact with Conductor (OSS and Orkes) REST APIs.
- 🔗 **Composio** - Use **Composio** to connect 100+ tools. Zero setup. Auth built-in. Made for agents, works for humans.
- 🔗 **Confluent** - Interact with Confluent Kafka and Confluent Cloud REST APIs.
- 🔗 **Contrast Security** - Brings Contrast's vulnerability and SCA data into your coding agent to quickly remediate vulnerabilities.
- 🔗 **Convex** - Introspect and query your apps deployed to Convex.
- 🔗 **Cortex** - Official MCP server for **Cortex**.
- 🔗 **Couchbase** - Interact with the data stored in Couchbase clusters.
- 🔗 **CRIC Wuye AI** - Interact with capabilities of the CRIC Wuye AI platform, an intelligent assistant specifically for the property management industry.
- 🔗 **CrowdStrike Falcon** - Connects AI agents with the CrowdStrike Falcon platform for intelligent security analysis, providing programmatic access to detections, incidents, behaviors, threat intelligence, hosts, vulnerabilities, and identity protection capabilities.
- 🔗 **CTERA Edge Filer** - CTERA Edge Filer delivers intelligent edge caching and multiprotocol file access, enabling fast, secure access to files across core and remote sites.
- 🔗 **CTERA Portal** - CTERA Portal is a multi-tenant, multi-cloud platform that delivers a global namespace and unified management across petabytes of distributed content.
- 🔗 **Cyccode** - Boost security in your dev lifecycle via SAST, SCA, Secrets & IaC scanning with **Cyccode**.
- 🔗 **Dart** - Interact with task, doc, and project data in **Dart**, an AI-native project management tool
- 🔗 **Databricks** - Connect to data, AI tools & agents, and the rest of the Databricks platform using turnkey managed MCP servers. Or, host your own custom MCP servers within the Databricks security and data governance boundary.
- 🔗 **DataHub** - Search your data assets, traverse data lineage, write SQL queries, and more using **DataHub** metadata.
- 🔗 **Daytona** - Fast and secure execution of your AI generated code with **Daytona** sandboxes
- 🔗 **Debugg.AI** - Zero-Config, Fully AI-Managed End-to-End Testing for any code gen platform via **Debugg.AI** remote browsing test agents.
- 🔗 **DeepL** - Translate or rewrite text with **DeepL**'s very own AI models using **the DeepL API**
- 🔗 **Defang** - Deploy your project to the cloud seamlessly with the **Defang** platform without leaving your integrated development environment
- 🔗 **Detailer** - Instantly generate rich, AI-powered documentation for your GitHub repositories. Designed for AI agents to gain deep project context before taking action.
- 🔗 **DevCycle** - Create and monitor feature flags using natural language in your AI coding assistant.
- 🔗 **DevHub** - Manage and utilize website content within the **DevHub** CMS platform
- 🔗 **DevRev** - An MCP server to integrate with DevRev APIs to search through your DevRev Knowledge Graph where objects can be imported from diff. Sources listed **here**.
- 🔗 **DexPaprika (CoinPaprika)** - Access real-time DEX data, liquidity pools, token information, and trading analytics across multiple blockchain networks with **DexPaprika** by CoinPaprika.
- 🔗 **Dolt** - The official MCP server for version-controlled **Dolt** databases.
- 🔗 **Dot (GetDot.ai)** - Fetch, analyze or visualize data from your favorite database or data warehouse (Snowflake, BigQuery, Redshift, Databricks, Clickhouse, ...) with **Dot**, your AI Data Analyst. This remote MCP server is a one-click integration for user that have setup Dot.
- 🔗 **Drata** - Get hands-on with our experimental MCP server—bringing real-time compliance intelligence into your AI workflows.
- 🔗 **Dumpling AI** - Access data, web scraping, and document conversion APIs by **Dumpling AI**

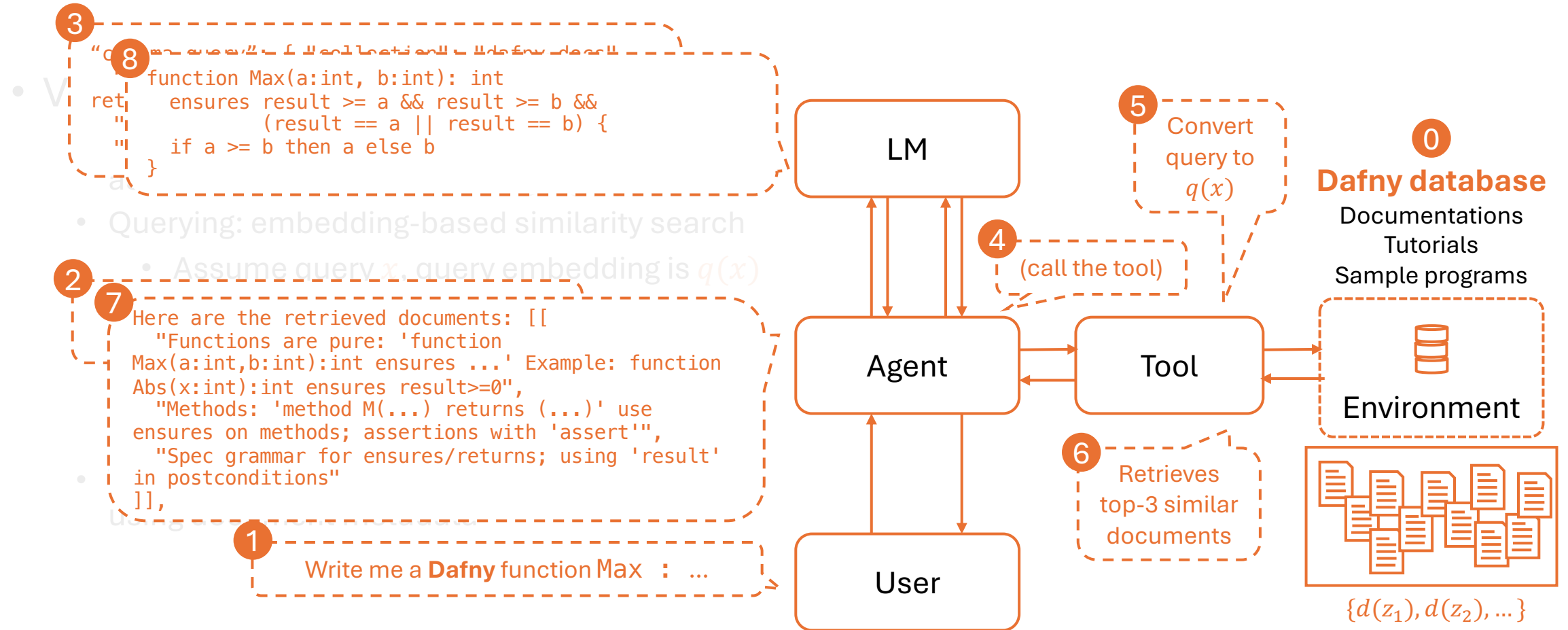
- 🔗 **GitGuardian** - GitGuardian official MCP server - Scan projects using GitGuardian's industry-leading API, which features over 500 secret detectors to prevent credential leaks before they reach public repositories. Resolve security incidents directly with rich contextual data for rapid, automated remediation.
- 🔗 **GitHub** - GitHub's official MCP Server.
- 🔗 **GitKraken** - A CLI for interacting with GitKraken APIs. Includes an MCP server via **gk mcp** that not only wraps GitKraken APIs, but also Jira, GitHub, GitLab, and more.
- 🔗 **Glean** - Enterprise search and chat using Glean's API.
- 🔗 **Globalping** - Access a network of thousands of probes to run network commands like ping, traceroute, mtr, http and DNS resolve.
- 🔗 **gNucleus Text-To-CAD** - Generate CAD parts and assemblies from text using gNucleus AI models.
- 🔗 **Google Cloud Run** - Deploy code to Google Cloud Run
- 🔗 **GoLogin MCP server** - Manage your GoLogin browser profiles and automation directly through AI conversations!
- 🔗 **Google Maps Platform Code Assist** - Ground agents on fresh, official documentation and code samples for optimal geo-related guidance and code..
- 🔗 **gotoHuman** - Human-in-the-loop platform - Allow AI agents and automations to send requests for approval to your **gotoHuman** inbox.
- 🔗 **Grafana** - Search dashboards, investigate incidents and query datasources in your Grafana instance
- 🔗 **Grafbase** - Turn your GraphQL API into an efficient MCP server with schema intelligence in a single command.
- 🔗 **Grain** - Access your Grain meetings notes & transcripts directly in claude and generate reports with native Claude Prompts.
- 🔗 **Graphlit** - Ingest anything from Slack to Gmail to podcast feeds, in addition to web crawling, into a searchable **Graphlit** project.
- 🔗 **Gremlin** - The official **Gremlin** MCP server. Analyze your reliability posture, review recent tests and chaos engineering experiments, and create detailed reports.
- 🔗 **GreptimeDB** - Provides AI assistants with a secure and structured way to explore and analyze data in **GreptimeDB**.
- 🔗 **GROWI** - Official MCP Server to integrate with GROWI APIs.
- 🔗 **Gyazo** - Search, fetch, upload, and interact with Gyazo images, including metadata and OCR data.
- 🔗 **Harper** - An MCP server providing an interface for MCP clients to access data within **Harper**.
- 🔗 **Heroku** - Interact with the Heroku Platform through LLM-driven tools for managing apps, add-ons, dynos, databases, and more.
- 🔗 **HeyOnCall** - Page a human, sending critical or non-critical alerts to the free **HeyOnCall** iOS or Android apps.
- 🔗 **Hiveflow** - Create, manage, and execute agentic AI workflows directly from your assistant.
- 🔗 **Hive Intelligence** - Ultimate cryptocurrency MCP for AI assistants with unified access to crypto, DeFi, and Web3 analytics
- 🔗 **Hologres** - Connect to a **Hologres** instance, get table metadata, query and analyze data.
- 🔗 **Homebrew** Allows **Homebrew** users to run Homebrew commands locally.
- 🔗 **Honeycomb** Allows **Honeycomb** Enterprise customers to query and analyze their data, alerts, dashboards, and more; and cross-reference production behavior with the codebase.
- 🔗 **HubSpot** - Connect, manage, and interact with **HubSpot** CRM data
- 🔗 **Hugging Face** - Connect to the Hugging Face Hub APIs programmatically: semantic search for spaces and papers, exploration of datasets and models, and access to all compatible MCP Gradio tool spaces!
- 🔗 **Hunter** - Interact with the **Hunter API** to get B2B data using natural language.
- 🔗 **Hyperbolic** - Interact with Hyperbolic's GPU cloud, enabling agents and LLMs to view and rent available GPUs, SSH into them, and run GPU-powered workflows for you.
- 🔗 **Hyperbrowser - Hyperbrowser** is the next-generation platform empowering AI agents and enabling effortless, scalable browser automation.
- 🔗 **IBM wxflows** - Tool platform by IBM to build, test and deploy tools for any data source
- 🔗 **Inbox Zero** - AI personal assistant for email **Inbox Zero**
- 🔗 **Inflectra Spira** - Connect to your instance of the SpiraTest, SpiraTeam or SpiraPlan application lifecycle management platform by **Inflectra**
- 🔗 **Inkeep** - RAG Search over your content powered by **Inkeep**
- 🔗 **Integration App** - Interact with any other SaaS applications on behalf of your customers.
- 🔗 **IP2Location.io** - Interact with IP2Location.io API to retrieve the geolocation information for an IP address.
- 🔗 **IPLocate** - Look up IP address geolocation, network information, detect proxies and VPNs, and find abuse contact details using **IPLocate.io**

# File System (FS) for Code Implementation

- Create, read, write, or edit files

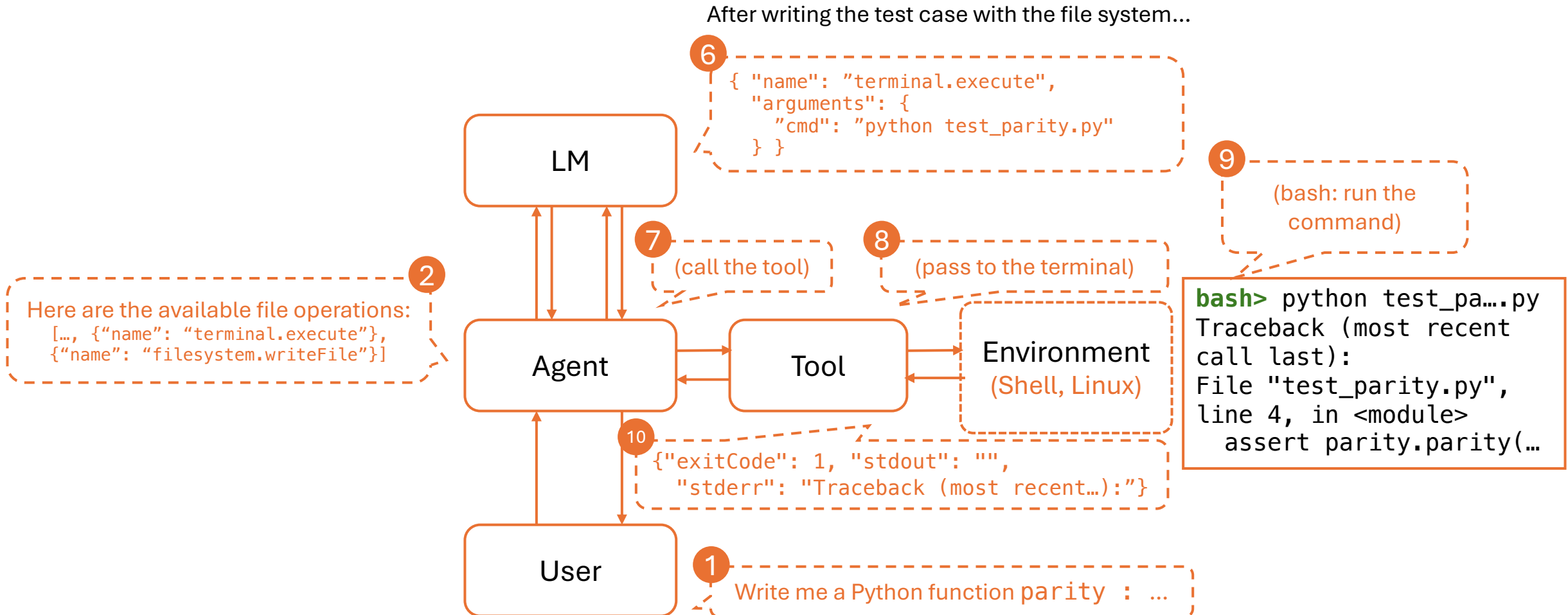


# Retrieval Augmented Generation (RAG)

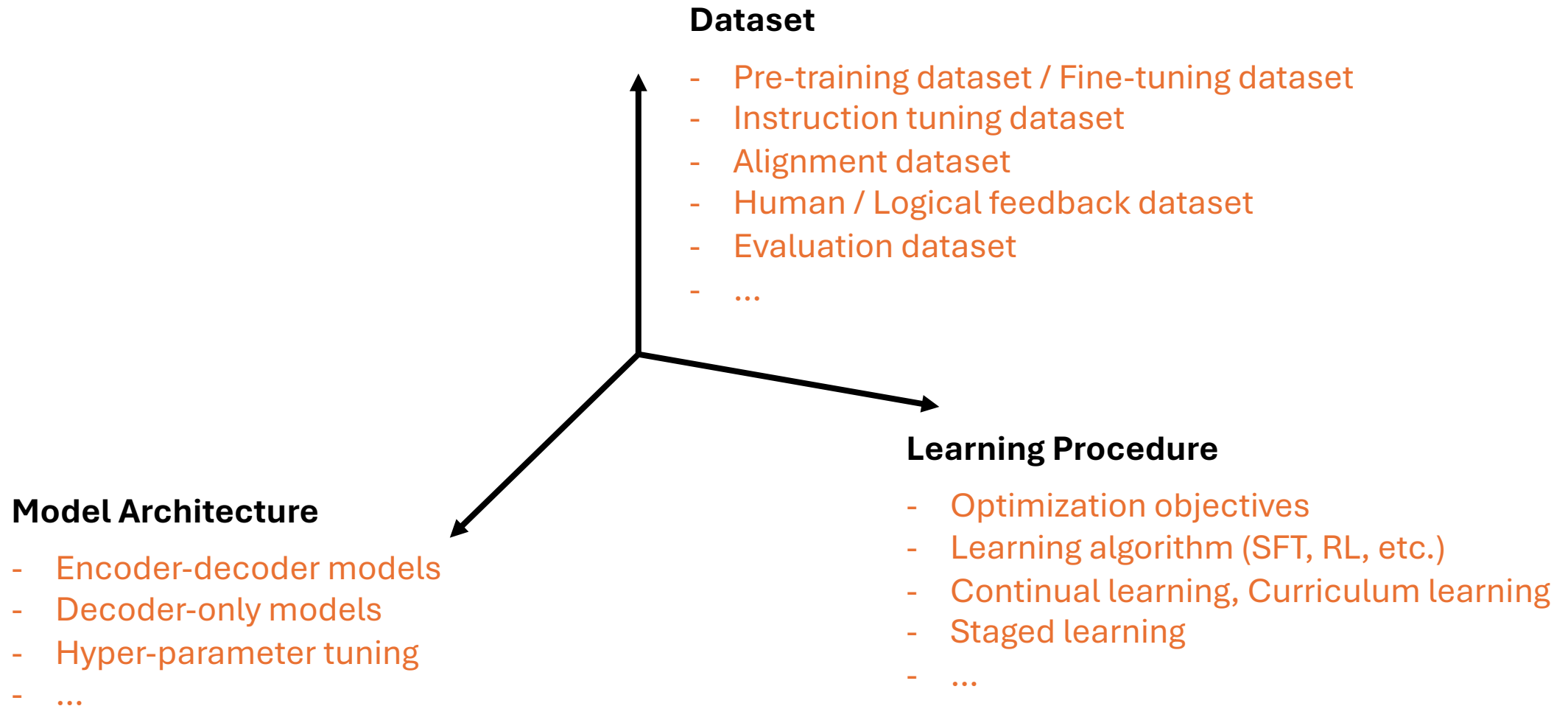




# Terminal as Tool

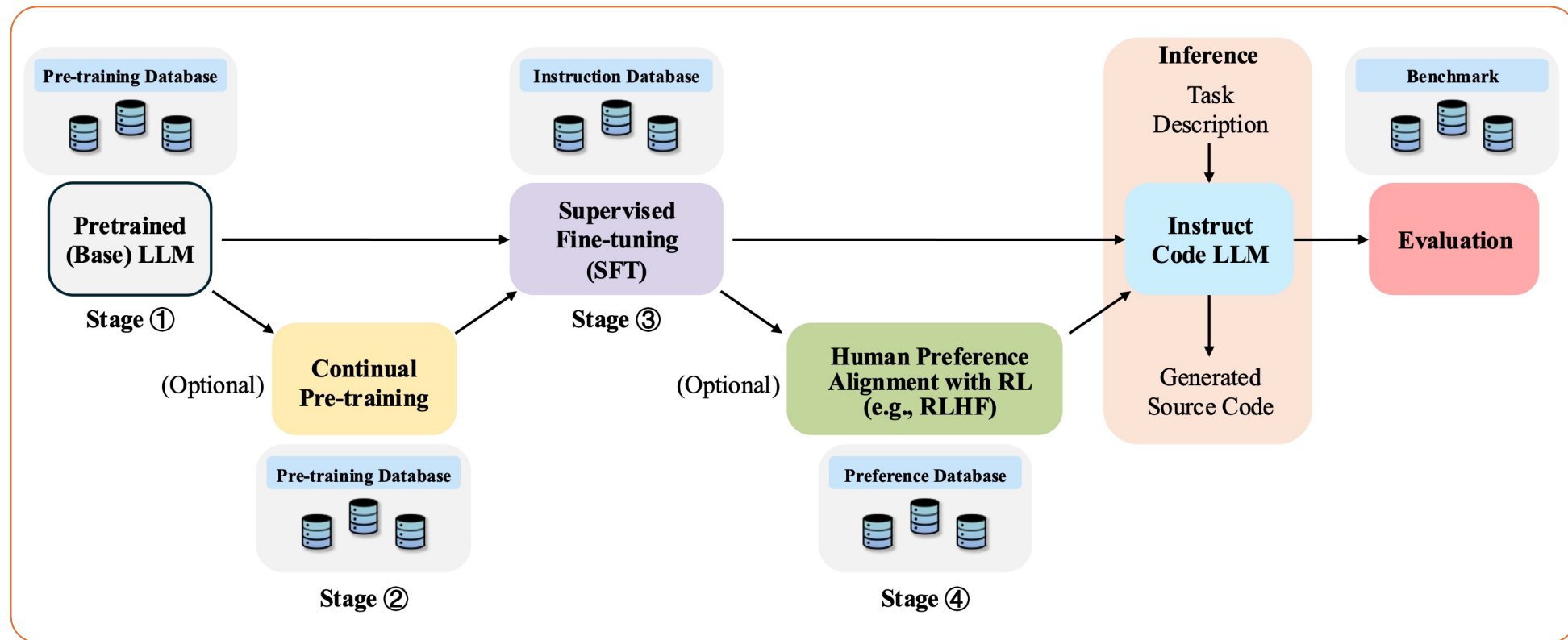


# Pre-training & Post-training of LLMs





# High-level Training, Inference, and Evaluation



# Pre-training: Optimizations

$$\theta = \operatorname{argmin}_{\theta} \sum_{\mathbf{x} \in \mathcal{D}} \mathcal{L}(\mathbf{x}; \theta)$$

- How do we find the “optimal” set of parameters  $\theta$ ?
  - Back-propagation
  - Optimizers
  - Learning rates & schedulers
  - **Batching & parallelism**

Gradient (at step  $t$ ):  $g_t = \nabla_{\theta} \mathcal{L}(\theta_t)$

Optimizer: SGD, Adam, **AdamW**, Momentum

First momentum:  $m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$

Second momentum:  $v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$

Weight Update:  $\theta_{t+1} = \theta_t - \eta \left( \frac{\hat{m}_t}{\sqrt{\hat{v}_t + \epsilon}} + \lambda \theta_t \right)$

Learning Rate Scheduling:  $\eta_t$

Cosine decay:  $\eta_t = \eta_{\min} + 0.5(\eta_{\max} - \eta_{\min})(1 + \cos(\frac{\pi t}{T}))$

Batching:  $B \subset D$

Mini-batched gradient:  $g_t = \frac{1}{|B|} \sum_{\mathbf{x} \in B} \nabla_{\theta} \mathcal{L}(\mathbf{x}; \theta_t)$

# SFT: Learning Objectives

- Pre-training: General understanding of language
- SFT: Aligns with human intent
- High-level Objective:
  - Instruction following / dialog
  - Reasoning and chain-of-thought
  - Tool use and agentic protocol-following
  - Code generation (completion + Infilling/FIM)
- Low-level Objective: Next-token prediction

$$\mathcal{L}(\mathbf{x}; \theta) = \sum_{i=1}^n -\log P_{\theta}(x_i | \mathbf{x}_{<i})$$

# RL: Optimization

- **DQN** (Deep Q-Network)

- Directly model Q-value: the value function of an action  $Q_{\theta}(s, a)$

- **Policy Gradient** (REINFORCE)

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\pi_{\theta}} [\nabla_{\theta} \log \pi_{\theta}(a|s) R]$$

- Directly optimize the policy  $\pi_{\theta}(a|s)$

- **Actor-Critic Model** (AC)

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\pi_{\theta}} [\nabla_{\theta} \log \pi_{\theta}(a|s) A(s, a)]$$

- Adding a critic  $V_{\phi}(s)$  and “advantage estimate”  $A(s, a) = R - V_{\phi}(s)$

- **Proximal Policy Optimization** (PPO)  $\nabla_{\theta} J(\theta) = \mathbb{E}_{\pi_{\theta}} [\nabla_{\theta} \log \pi_{\theta}(y|x) (r_{\phi}(x, y) - \beta(\log \pi_{\theta}(y|x) -$

- To stabilize, we clip the loss based on how far the policy deviates

- **Group Relative Policy Optimization** (GRPO)

- (Specifically for LLM) grouped normalized advantage  $A_i = (r_i - \bar{r}) / \sigma_r$

# Forms of Software Analysis

## Static Analysis

Examine code without concrete execution

- Taint analysis
- Reachability analysis
- Abstract interpretation
- Symbolic execution
- ...

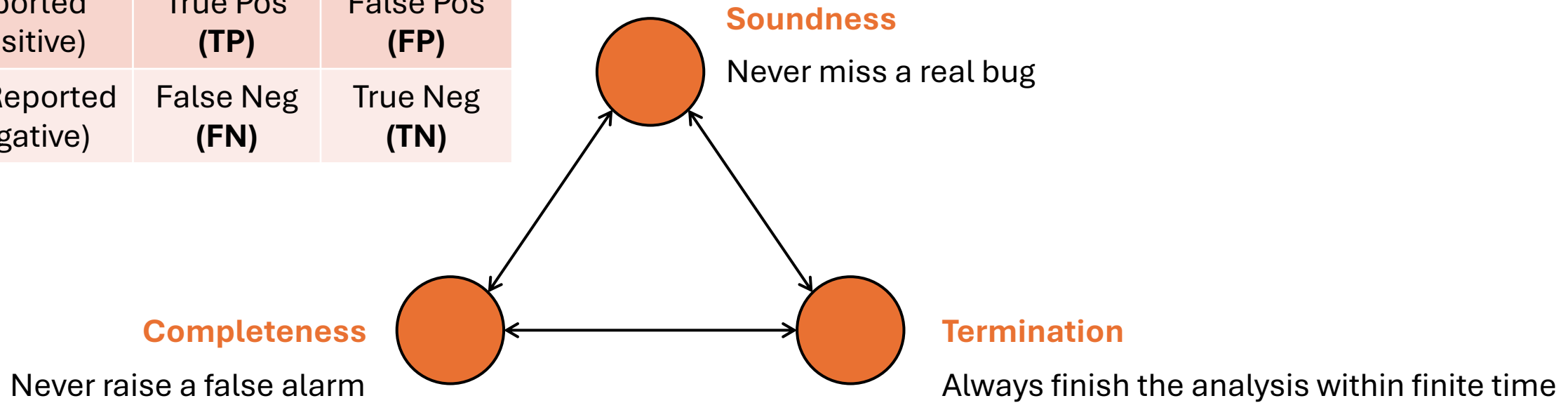
## Dynamic Analysis

Observes program behavior while executing

- Unit testing
- Fuzzing
- Property-based testing
- Penetration testing
- ...

# Software Analysis: Impossible Triangle

|                         | Real Bug       | Not a Bug      |
|-------------------------|----------------|----------------|
| Reported (positive)     | True Pos (TP)  | False Pos (FP) |
| Not Reported (negative) | False Neg (FN) | True Neg (TN)  |



# Dynamic Analysis

Human Effort → LLM: write the **input generator** and **fuzzing harness**

Unit Testing

Human-written tests  
in the form of concrete  
input-output

```
assert f(2) == 4
```

Fuzzing

Randomly  
generating  
test inputs

```
for i in 0..10^4:  
    x = gen_input()  
    f(x)
```

Human Effort → LLM: figure out  
**input-outputs**, write the **test cases**

Human Effort → LLM: figure out the **very bad input**, describe the **bad behavior**

Property-based testing

Randomly generating  
test inputs, but with  
properties checking outputs

```
for i in 0..10^4:  
    x = gen_input()  
    y = f(x)  
    assert property(x,y)
```

Human Effort → LLM: write the **input generator**, **harness** and the **property**

Penetration testing

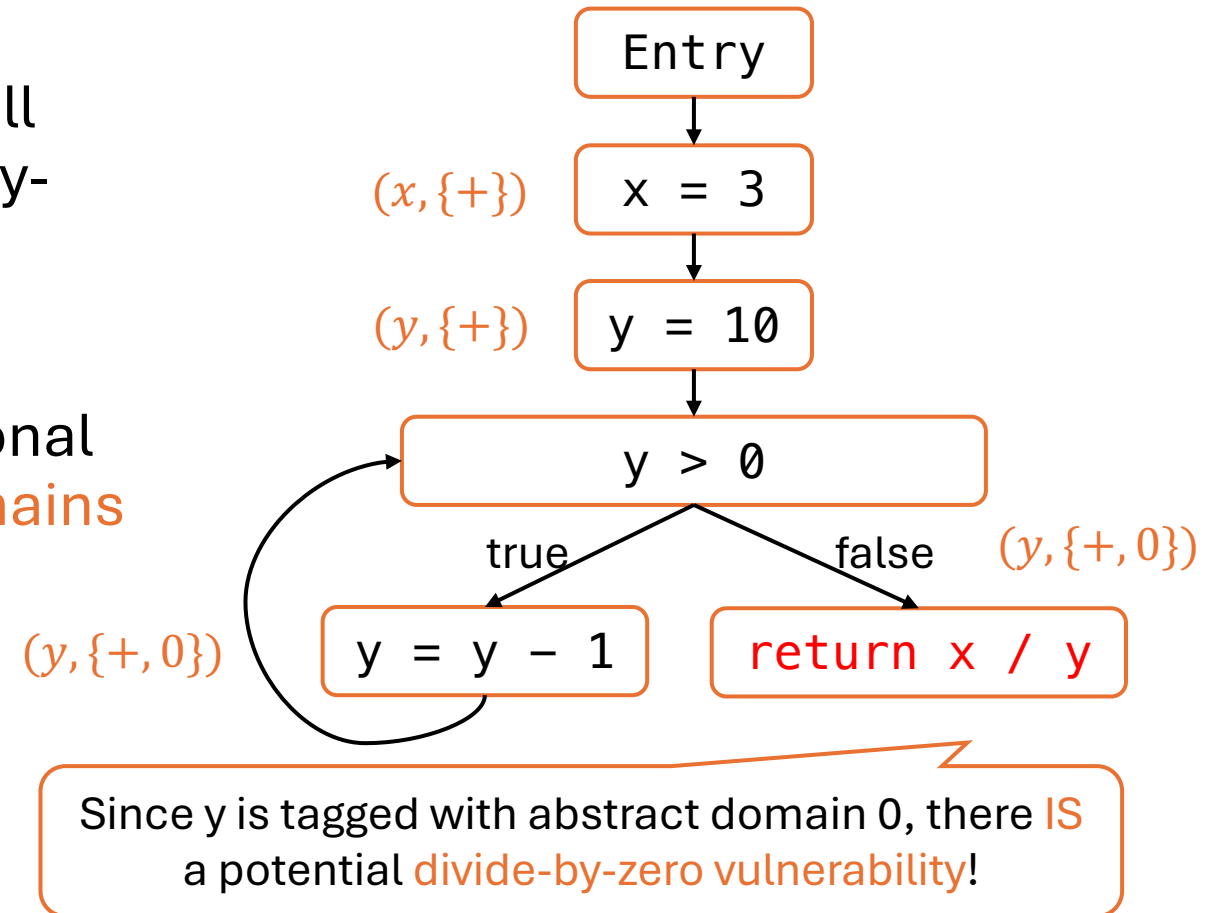
Crafting adversarial inputs  
to trigger vulnerabilities rather  
than getting outputs

```
try:  
    f(very_bad_input)  
    assert false  
except:  
    assert no_bad_behavior
```

# Static Dataflow Analysis

- Question:
  - With Dataflow analysis, can we tell whether there could be a divide-by-zero vulnerability?
- Answer:
  - We can tag the nodes with additional information such as **abstract domains**

Abstract Domain:  $\{+, 0, -\}$





# Language-based Security: Who can be trusted?

## C/C++ Program

```
int main() {  
    int *arr1 = (int *)  
        malloc(100 * sizeof(int));  
    int *arr2 = arr1;  
    free(arr1);  
    arr2[3] = 27;  
    printf("%d\n", arr2[3]);  
}
```

Developer / LLM

**NO**

## Python Program

```
def main():  
    arr1 = [()] * 100  
    arr2 = arr1  
    del arr1  
    arr2[3] = 27  
    print(arr2[3])
```

### Python Runtime

Memory Management  
Reference Counting  
Garbage Collection

Maybe yes

## Rust Program

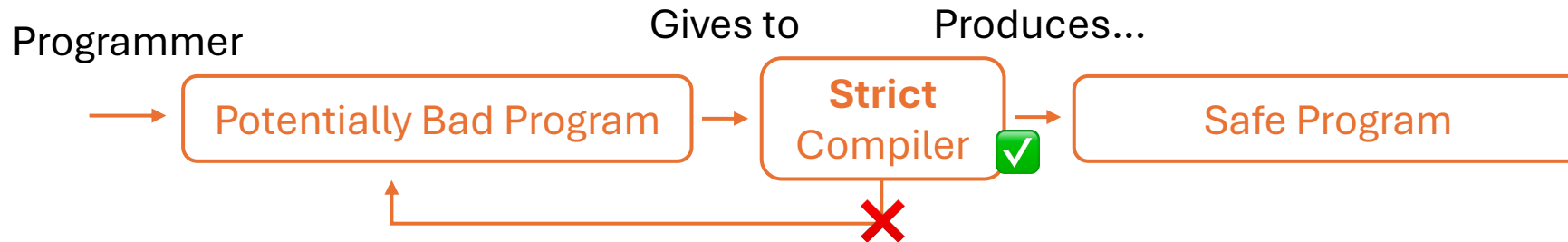
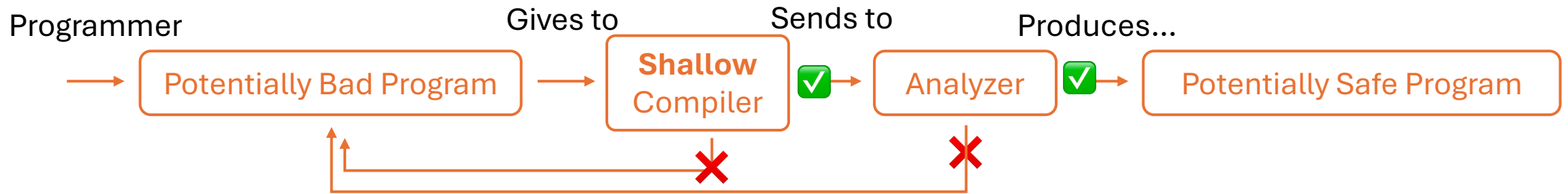
```
fn main() {  
    let mut arr1 = vec![0; 100];  
    let arr2 = &mut arr1;  
    arr2[3] = 27;  
    arr1[2] = 30;  
}
```

### Rust Compiler

Linear type system  
Ownership & borrow checker  
Life-time resolver

Maybe yes

# Synthesizing Safe-by-Construct Programs



# Applications

# Advanced Topics & Applications

- Topics

- Reinforcement Learning
- Evolutionary Synthesis
- Multi-agent Frameworks

- Applications

- Security Red- and Blue-Teaming, Program Analysis
- Software Engineering, Testing, Optimization, Transpilation
- Planning & Cyber-Physical System
- Math & Theorem Proving

# Special Topics

# Special Topics

Hardware Accelerated Computation (CUDA)

# Synthesizing CUDA Kernels

KernelBench: Can LLMs Write Efficient GPU Kernels?

Anne Ouyang<sup>1,\*</sup>, Simon Guo<sup>1,\*</sup>, Simran Arora<sup>1</sup>, Alex L. Zhang<sup>2</sup>, William Hu<sup>1</sup>, Christopher Ré<sup>1</sup>, and Azalia Mirhoseini<sup>1</sup>

<sup>1</sup>Stanford University

<sup>2</sup>Princeton University

February 18, 2025

# KernelBench: Can LLMs Write Efficient GPU Kernels?

Anne Ouyang<sup>1,\*</sup>, Simon Guo<sup>1,\*</sup>, Simran Arora<sup>1</sup>, Alex L. Zhang<sup>2</sup>, William Hu<sup>1</sup>, Christopher Ré<sup>1</sup>, and Azalia Mirhoseini<sup>1</sup>

<sup>1</sup>Stanford University

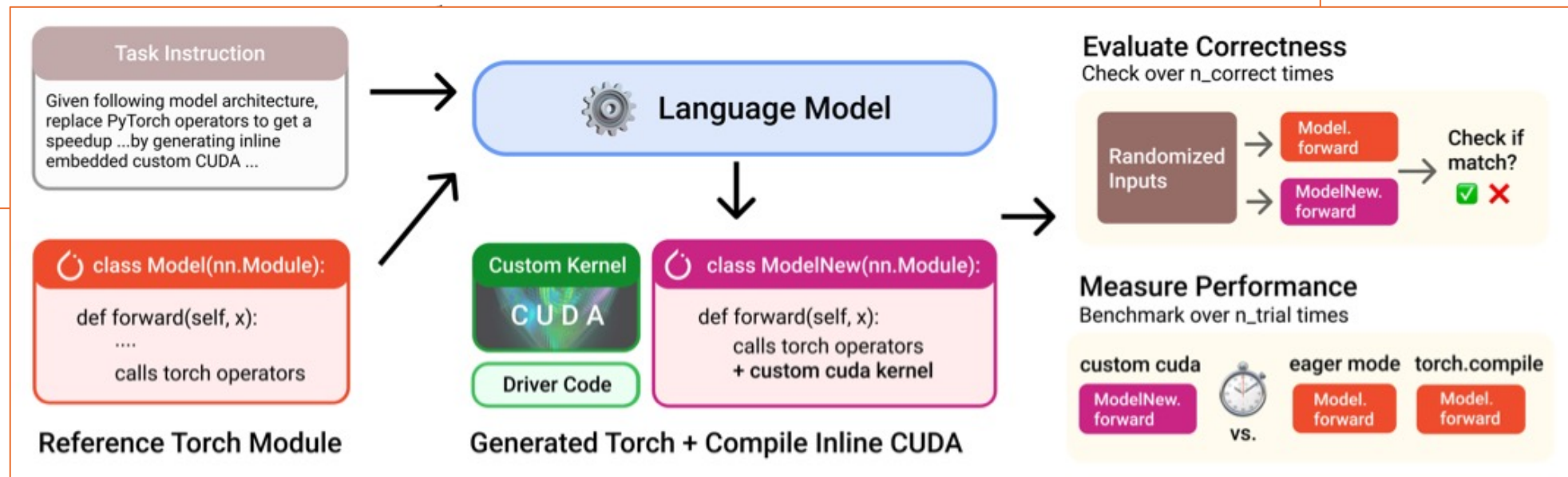
<sup>2</sup>Princeton University

February 18, 2025



# KernelBench: Can LLMs Write Efficient GPU Kernels?

Anne Ouyang<sup>1,\*</sup>, Simon Guo<sup>1,\*</sup>, Simran Arora<sup>1</sup>, Alex L. Zhang<sup>2</sup>, William Hu<sup>1</sup>, Christopher Ré<sup>1</sup>, and Azalia Mirhoseini<sup>1</sup>



Anne

# KernelBench: C

## Task Instruction

Given following model architecture, replace PyTorch operators to get a speedup ...by generating inline embedded custom CUDA ...

 **class Model(nn.Module):**

**def forward(self, x):**

....

calls torch operators

## Reference Torch Module

```
1 import torch
2 import torch.nn as nn
3 from torch.utils.cpp_extension import load_inline
4
5 # Define the custom CUDA kernel for matrix multiplication
6 matmul_source = """
7 #include <torch/extension.h>
8 #include <cuda_runtime.h>
9
10 __global__ void matmul_kernel(const float* A, const float* B, float* C, int M, int N, int K)
11     ↪ {
12     int row = blockIdx.y * blockDim.y + threadIdx.y;
13     int col = blockIdx.x * blockDim.x + threadIdx.x;
14
15     if (row < M && col < N) {
16         float sum = 0.0f;
17         for (int k = 0; k < K; ++k) {
18             sum += A[row * K + k] * B[k * N + col];
19         }
20         C[row * N + col] = sum;
21     }
22 }
23
24 torch::Tensor matmul_cuda(torch::Tensor A, torch::Tensor B) {
25     int M = A.size(0);
26     int N = B.size(1);
27     int K = A.size(1);
28
29     auto C = torch::zeros({M, N}, A.options());
30
31     dim3 threadsPerBlock(16, 16);
32     dim3 numBlocks((N + threadsPerBlock.x - 1) / threadsPerBlock.x, (M + threadsPerBlock.y -
33     ↪ 1) / threadsPerBlock.y);
34
35     matmul_kernel<<<numBlocks, threadsPerBlock>>>(A.data_ptr<float>(), B.data_ptr<float>(),
36     ↪ C.data_ptr<float>(), M, N, K);
37
38     return C;
39 }
40 """
41
42 matmul_cpp_source = (
43     "torch::Tensor matmul_cuda(torch::Tensor A, torch::Tensor B);"
44 )
45
46 # Compile the inline CUDA code for matrix multiplication
47 matmul = load_inline(
48     name="matmul",
49     cpp_sources=matmul_cpp_source,
50     cuda_sources=matmul_source,
51     functions=["matmul_cuda"],
52     verbose=True,
53     extra_cflags=[""],
54     extra_ldflags=[""],
55 )
56
57 class ModelNew(nn.Module):
58     def __init__(self):
59         super(ModelNew, self).__init__()
60         self.matmul = matmul
61
62     def forward(self, A: torch.Tensor, B: torch.Tensor) -> torch.Tensor:
63         return self.matmul.matmul_cuda(A, B)
```

ness  
mes

Model.  
forward

Check if  
match?

ModelNew.  
forward



ance  
times

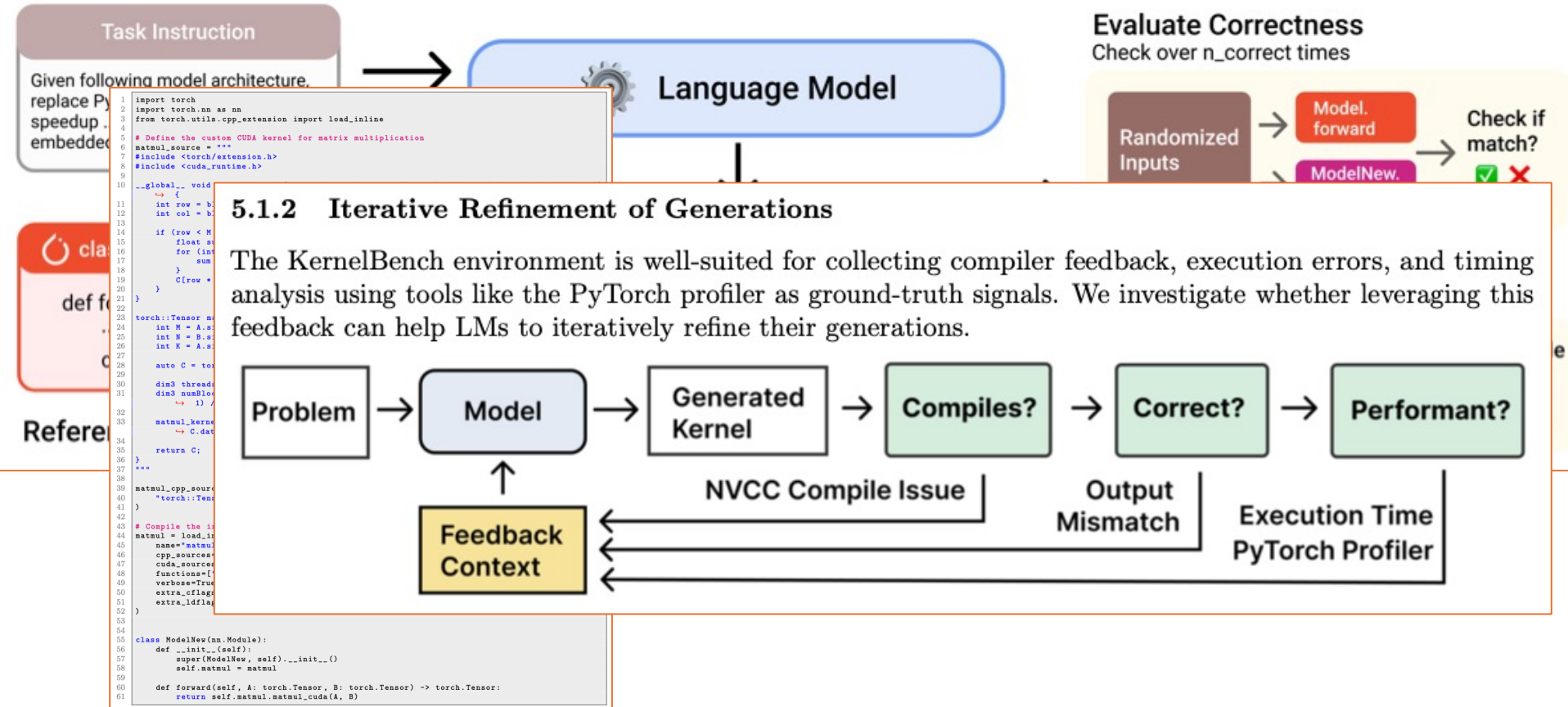
eager mode torch.compile

Model.  
forward

Model.  
forward

# KernelBench: Can LLMs Write Efficient GPU Kernels?

Anne







# KernelBench: Can LLMs Write Efficient GPU Kernels?

Appendix

| <b>fast<sub>1</sub></b> over: | PyTorch Eager |            |            | torch.compile |            |           |
|-------------------------------|---------------|------------|------------|---------------|------------|-----------|
| KernelBench Level             | 1             | 2          | 3          | 1             | 2          | 3         |
| GPT-4o                        | 4%            | 5%         | 0%         | 18%           | 4%         | 4%        |
| OpenAI o1                     | <u>10%</u>    | <u>24%</u> | <b>12%</b> | 28%           | <u>19%</u> | 4%        |
| DeepSeek V3                   | 6%            | 4%         | <u>8%</u>  | 20%           | 2%         | <u>2%</u> |
| DeepSeek R1                   | <b>12%</b>    | <b>36%</b> | 2%         | <b>38%</b>    | <b>37%</b> | <u>2%</u> |
| Claude 3.5 Sonnet             | <u>10%</u>    | 7%         | 2%         | <u>29%</u>    | 2%         | <u>2%</u> |
| Llama 3.1-70B Inst.           | 3%            | 0%         | 0%         | 11%           | 0%         | 0%        |
| Llama 3.1-405B Inst.          | 3%            | 0%         | 2%         | 16%           | 0%         | 0%        |

Table 1: **KernelBench is a challenging benchmark for current LMs.** Here we present fast<sub>1</sub>, i.e. the percentage of problems where the model-generated kernel is faster than the PyTorch Eager and torch.compile baseline (default configuration) on NVIDIA L40S.

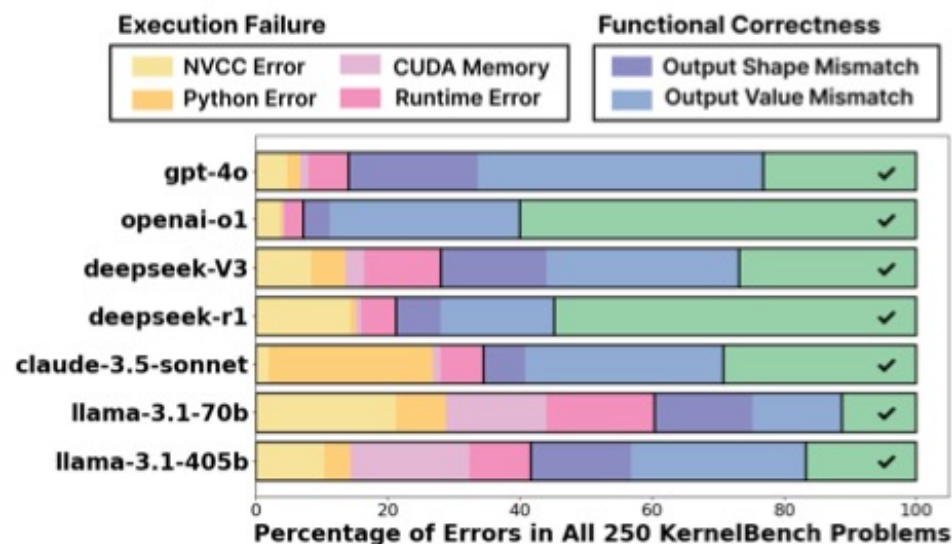


Figure 2: **We categorize failure modes of kernel code into execution failure and functional correctness.** For the one-shot baseline, reasoning models generate fewer kernels with execution failures, but all models struggle similarly with functional correctness.

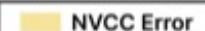
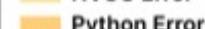
st<sub>p</sub>, which is  
o of PyTorch

which the LM

# KernelBench: Can LLMs Write Efficient GPU Kernels?

Appendix

|                               |               |   |   |               |   |   |
|-------------------------------|---------------|---|---|---------------|---|---|
| <b>fast<sub>1</sub></b> over: | PyTorch Eager |   |   | torch.compile |   |   |
| KernelBench Level             | 1             | 2 | 3 | 1             | 2 | 3 |

|                                                                                                  |                                                                                                   |                                                                                                           |  |
|--------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------|--|
| <b>Execution Failure</b>                                                                         |                                                                                                   | <b>Functional Correctness</b>                                                                             |  |
|  NVCC Error   |  CUDA Memory   |  Output Shape Mismatch |  |
|  Python Error |  Runtime Error |  Output Value Mismatch |  |

## 5.1.3 Comparing Repeated Sampling and Iterative Refinement

| Method                       | Level 1          |                |                | Level 2          |                |                | Level 3          |                |                |
|------------------------------|------------------|----------------|----------------|------------------|----------------|----------------|------------------|----------------|----------------|
|                              | Llama-3.1<br>70B | DeepSeek<br>V3 | Deepseek<br>R1 | Llama-3.1<br>70B | Deepseek<br>V3 | Deepseek<br>R1 | Llama-3.1<br>70B | Deepseek<br>V3 | Deepseek<br>R1 |
| Single Attempt (Baseline)    | 3%               | 6%             | 12%            | 0%               | 4%             | 36%            | 0%               | 8%             | 2%             |
| Repeated Sampling (@10)      | 5%               | 11%            | N/A            | 3%               | <b>14%</b>     | N/A            | 1%               | 14%            | N/A            |
| Iterative Refinement w G     | <b>9%</b>        | 9%             | 18%            | 0%               | 7%             | 44%            | 0%               | 14%            | 4%             |
| Iterative Refinement w G+E   | 5%               | 13%            | 41%            | <b>5%</b>        | 5%             | 62%            | <b>8%</b>        | <b>22%</b>     | 12%            |
| Iterative Refinement w G+E+P | 7%               | <b>19%</b>     | <b>43%</b>     | 4%               | 6%             | <b>72%</b>     | 2%               | 14%            | <b>18%</b>     |

ch is  
Torch

e LM

and `torch.compile` baseline (default configuration) on NVIDIA L40S.

but all models struggle similarly with functional correctness.

# FLASHATTENTION: Fast and Memory-Efficient Exact Attention with IO-Awareness

Tri Dao<sup>†</sup>, Daniel Y. Fu<sup>†</sup>, Stefano Ermon<sup>†</sup>, Atri Rudra<sup>‡</sup>, and Christopher Ré<sup>†</sup>

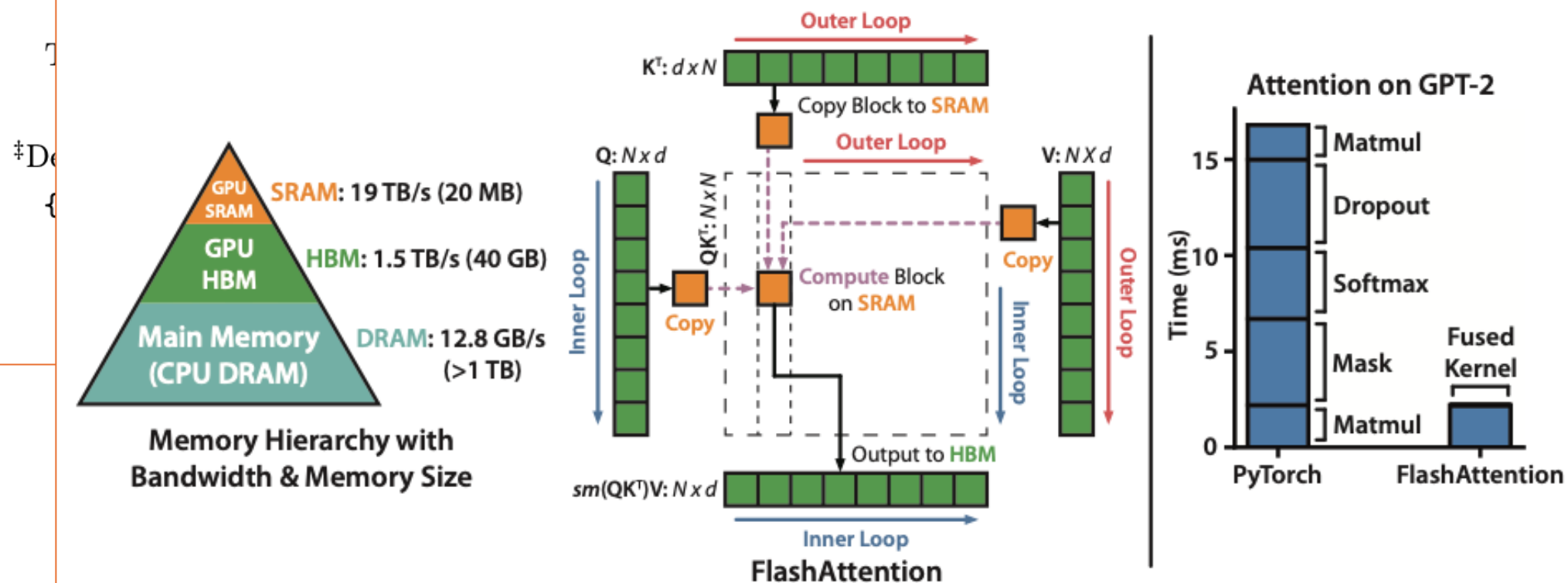
<sup>†</sup>Department of Computer Science, Stanford University

<sup>‡</sup>Department of Computer Science and Engineering, University at Buffalo, SUNY

`{trid,danfu}@cs.stanford.edu, ermon@stanford.edu, atri@buffalo.edu,`  
`chrismre@cs.stanford.edu`

June 24, 2022

# FLASHATTENTION: Fast and Memory-Efficient Exact Attention with IO Awareness





# FLASHATTENTION: Fast and Memory-Efficient Exact Attention with IO-Awareness

Tri Dao<sup>†</sup>, Daniel Y. Fu<sup>†</sup>, Stefano Ermon<sup>†</sup>, Atri Rudra<sup>‡</sup>, and Christopher Ré<sup>†</sup>

<sup>†</sup>Department of Computer Science, Stanford University

<sup>‡</sup>Department of Computer Science and Engineering, University at Buffalo, SUNY

{trid,danfu}@cs.stanford.edu, ermon@stanford.edu, atri@buffalo.edu,  
chrismre@cs.stanford.edu

June 24, 2022

## **QiMeng-Attention: SOTA Attention Operator is generated by SOTA Attention Algorithm**

**Qirui Zhou<sup>1,3</sup>, Shaohui Peng<sup>2</sup>, Weiqiang Xiong<sup>2,3</sup>, Haixin Chen<sup>1,3</sup>, Yuanbo Wen<sup>1</sup>,  
Haochen Li<sup>2</sup>, Ling Li<sup>2,3\*</sup>, Qi Guo<sup>1</sup>, Yongwei Zhao<sup>1</sup>, Ke Gao<sup>2</sup>, Ruizhi Chen<sup>2</sup>,  
Yanjun Wu<sup>2</sup>, Chen Zhao<sup>2</sup>, Yunji Chen<sup>1,3</sup>**

<sup>1</sup>SKL of Processors, Institute of Computing Technology, CAS, Beijing, China

<sup>2</sup>Intelligent Software Research Center, Institute of Software, CAS, Beijing China

<sup>3</sup>University of Chinese Academy of Sciences, Beijing, China

zhouqirui22s@ict.ac.cn, {pengshaohui, liling}@iscas.ac.cn

## **QiMeng-Attention: SOTA Attention Operator is generated by SOTA Attention Algorithm**

**Qirui Zhou<sup>1,3</sup>, Shaohui Peng<sup>2</sup>, Weiqiang Xiong<sup>2,3</sup>, Haixin Chen<sup>1,3</sup>, Yuanbo Wen<sup>1</sup>,  
Haochen Li<sup>2</sup>, Ling Li<sup>2,3\*</sup>, Qi Guo<sup>1</sup>, Yongwei Zhao<sup>1</sup>, Ke Gao<sup>2</sup>, Ruizhi Chen<sup>2</sup>,  
YanJun Wu<sup>2</sup>, Chen Zhao<sup>2</sup>, Yunji Chen<sup>1,3</sup>**

<sup>1</sup>SKL of Processors, Institute of Computing Technology, CAS, Beijing, China

<sup>2</sup>Intelligent Software Research Center, Institute of Software, CAS, Beijing China

<sup>3</sup>University of Chinese Academy of Sciences, Beijing, China

zhouqirui22s@ict.ac.cn, {pengshaohui, liling}@iscas.ac.cn

# QiMeng-Attention: SOTA

Qirui Zhou<sup>1,3</sup>, Shaohui Peng<sup>2</sup>, V  
Haochen Li<sup>2</sup>, Ling Li<sup>2,3\*</sup>, Qi G  
Yanjuan Wu<sup>2</sup>

<sup>1</sup>SKL of Processors, Institute of

<sup>2</sup>Intelligent Software Research Center

<sup>3</sup>University of Chinese Academy of Sciences

zhouqirui22s@ict.ac.cn

**Prompt:** Generate an attention operator on A100 GPU with Torch

## High-level Code

```
scores = torch.matmul(q, k.transpose(-2, -1))
scores = scores/torch.sqrt(torch.tensor(d_k, dtype=torch.float32))
attention_weights = F.softmax(scores, dim=-1)
output = torch.matmul(attention_weights, v)
```



**Performance  
bottleneck:  
redundant global  
memory access**

## Corresponding hardware processing pipeline

L GEMM1 S L Softmax S L GEMM2 S

L Load data from global memory S Store data to global memory Computation

(a) LLMs generate high-level code by calling Torch functions

**Prompt:** Generate a fused CUDA kernel for attention on A100 GPU

## CUDA Kernel using Tensor Core

```
.....
// Load the tiles of Q, K, and V into shared memory
auto Q_tile = load_tile(smem_Q, block_shape, threadIdx.x);
.....
// Perform the matrix multiplication using Tensor Cores
auto O_tile = mma_op(Q_tile, K_tile, V_tile); .....
```



1. Inconsistent  
language usage  
2. Improper  
Tensor Core  
utilization  
3. ...

(b) LLMs cannot generate correct fused CUDA Kernel

TL Sketch

Reasoning

TL Code

LLM-TL

## CUDA Kernel using Tensor Core

```
.....
for (int n = 0; n < num_blocks; n++) {
// GEMM sQ, sK.T; get rS
cp_async_wait<1>();
__syncthreads();
auto tSsQ = smem_thr_copy_Q.partition_S(sQ);
auto tSsK = smem_thr_copy_K.partition_S(sK);
.....
```



**35.16x  
Speedup**



Corresponding hardware  
processing pipeline

L GEMM1  
Softmax  
GEMM2 S

(c) Our LLM-TL paradigm generates performant CUDA kernel

Figure 1: Our LLM-TL enables LLMs to generate high-performance Tensor Core kernel for attention.

# Special Topics

Graphics & Shader (GLSL, HLSL, WebGL, etc.)

ShaderGPT

Build GLSL shaders with natural language

Favorites

History

Explore

organic motion background

a kaleidoscope effect with trippy patterns and dreamscape with chromatic aberration effect

dreamscape

isometric vector art grid

sun

smoke mist turbulent

a smooth rainbow gradient with colors gently

soft rainbow gradients flowing into eachother

morphing madness

gentle rainbow gradient

swirling

displacement shader on image with distortion

Explore more

shadergpt.14islands.com

organic motion background

```
vec3 finalColor = mix(col1, col2, n1);
finalColor = mix(finalColor, col3, n2 * 0.5);
finalColor += n3 * 0.2;

finalColor += vec3(mouseEffect * 0.2);

gl_FragColor = vec4(finalColor, 1.0);

#include <colorspace_fragment>
}
```

Copy code ↗

organic motion background

↑

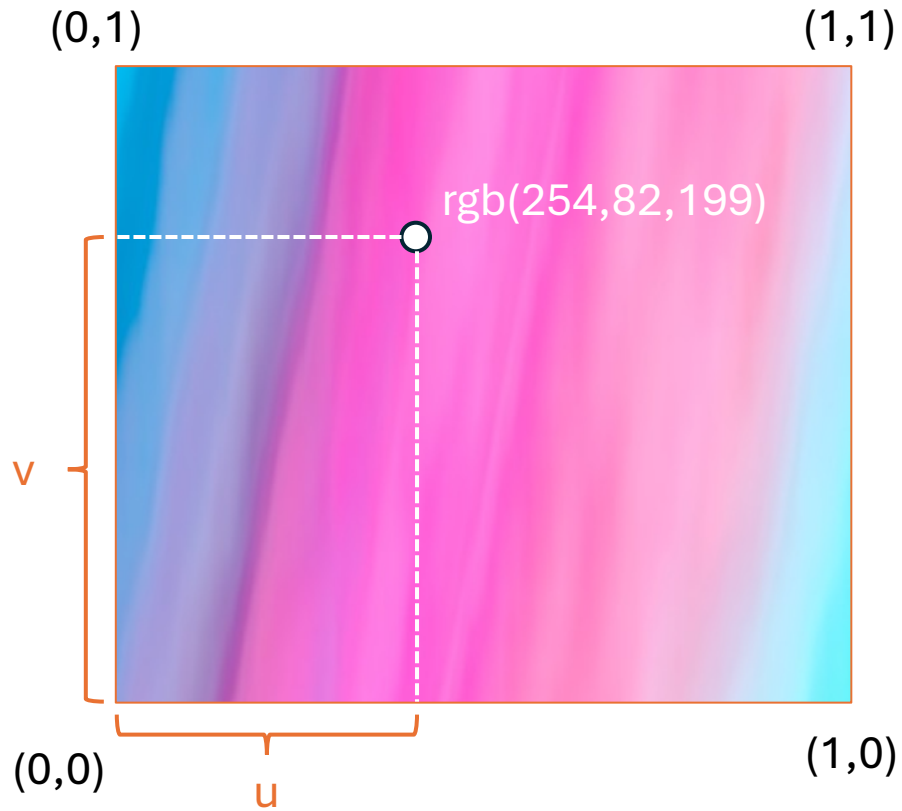
Daily limit 0/14

Temperature 0.5

AI Model 

claude-sonnet-3.7

# Shader



■ `gl_FragColor = vec3(0.99,0.32,0.78)`

```
vec2 uv = v_uv;
vec2 aspect = vec2(u_resolution.x/u_resolution.y, 1.0);
uv = uv * 2.0 - 1.0;
uv *= aspect;

vec2 mouseInfluence = (u_mouse * 2.0 - 1.0) * aspect;
float mouseDist = length(uv - mouseInfluence);
float mouseEffect = smoothstep(0.5, 0.0, mouseDist);

float t = u_time * 0.2;
vec2 movement = vec2(sin(t * 0.5), cos(t * 0.7));

float n1 = fbm(uv * 3.0 + movement + mouseEffect);
float n2 = fbm(uv * 2.0 - movement - mouseEffect);
float n3 = fbm(uv * 4.0 + vec2(n1, n2));

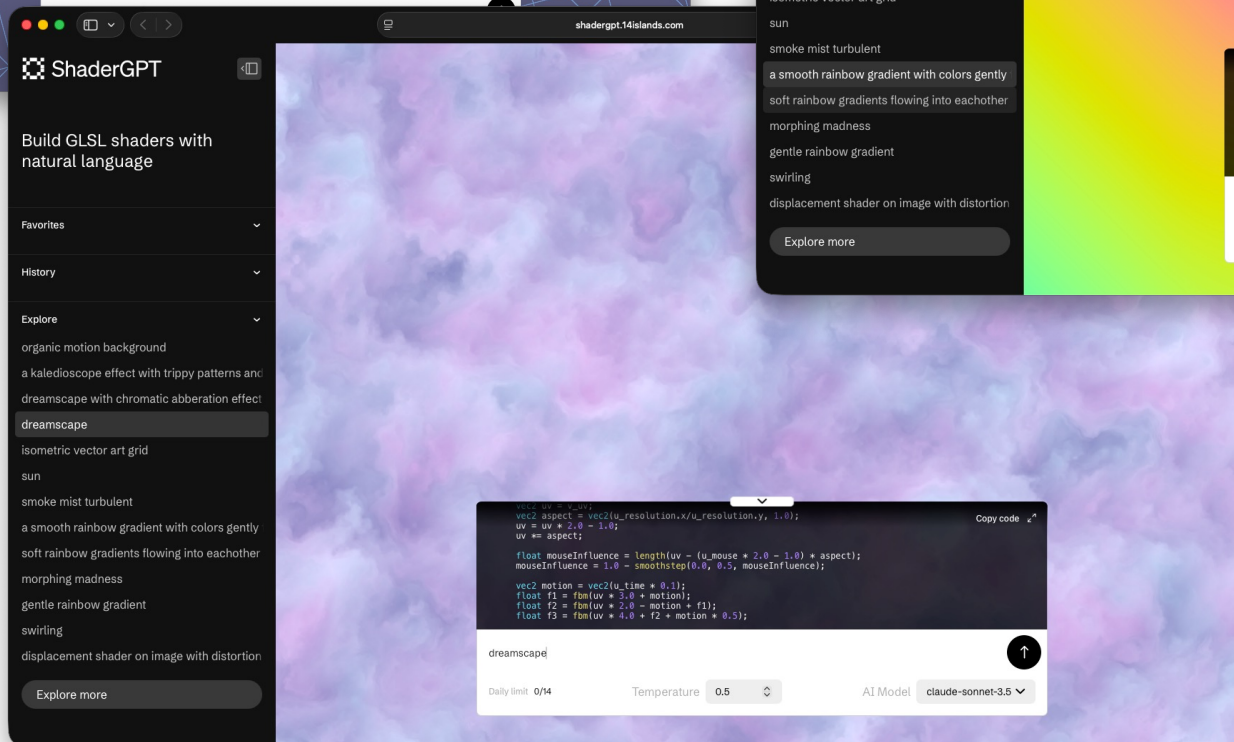
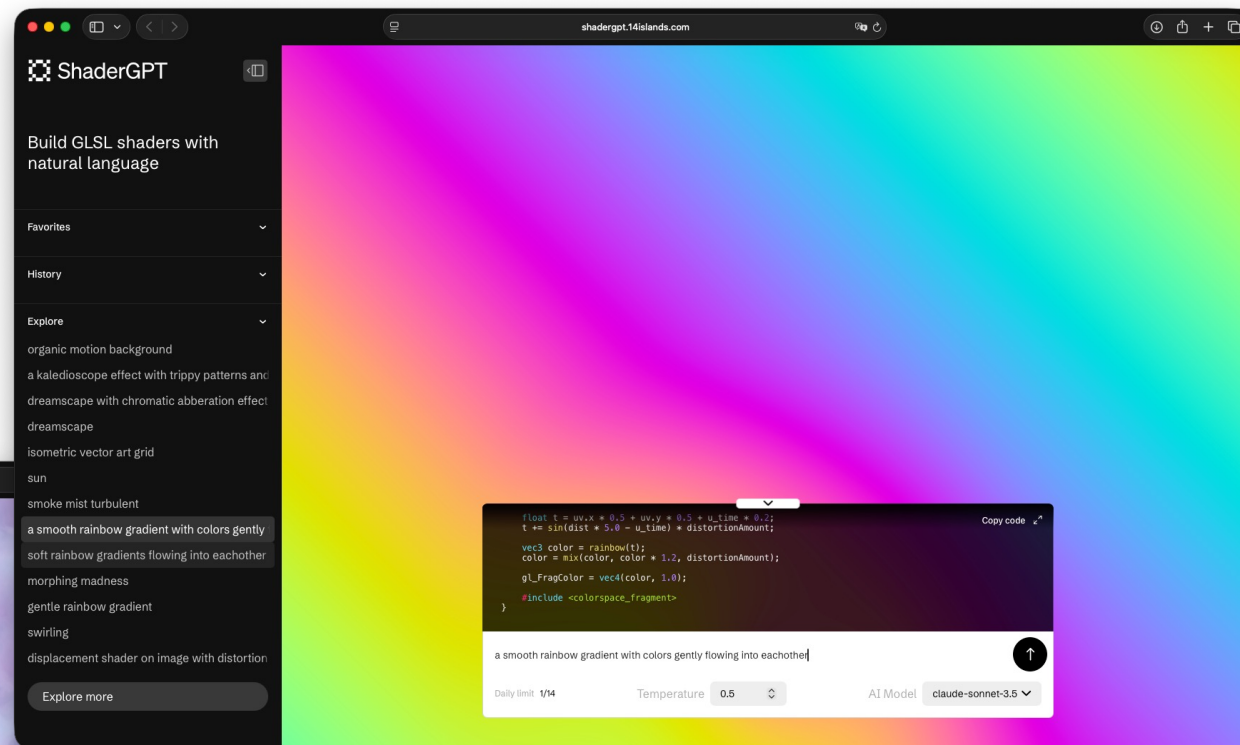
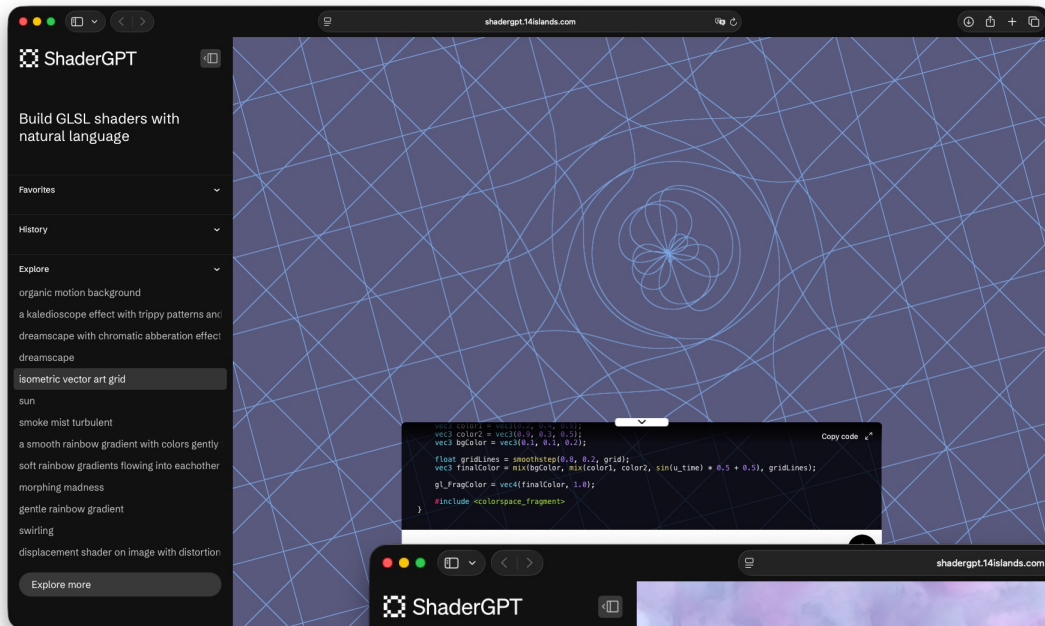
vec3 col1 = vec3(0.2, 0.5, 0.8);
vec3 col2 = vec3(0.8, 0.2, 0.5);
vec3 col3 = vec3(0.1, 0.8, 0.4);

vec3 finalColor = mix(col1, col2, n1);
finalColor = mix(finalColor, col3, n2 * 0.5);
finalColor += n3 * 0.2;

finalColor += vec3(mouseEffect * 0.2);
gl_FragColor = vec4(finalColor, 1.0);

#include <colorspace_fragment>
```





# Special Topics

Animations (2D/UI → 3D)



## LogoMotion: Visually-Grounded Code Synthesis for Creating and Editing Animation

Vivian Liu  
vivian@cs.columbia.edu  
Columbia University  
New York, NY, USA

Rubaiat Habib Kazi  
rhabib@adobe.com  
Adobe Research  
Seattle, WA, USA

Li-Yi Wei  
liyiwei@acm.org  
Adobe Research  
San Jose, CA, USA

Matthew Fisher  
matfishe@adobe.com  
Adobe Research  
San Francisco, CA, USA

Timothy Langlois  
tlangloi@adobe.com  
Adobe Research  
Seattle, WA, USA

Seth Walker  
swalker@adobe.com  
Adobe Research  
San Francisco, CA, USA

Lydia Chilton  
chilton@cs.columbia.edu  
Columbia University  
New York, NY, USA

# LogoMotion: Visually-Grounded Code Synthesis for Creating and Editing Animation

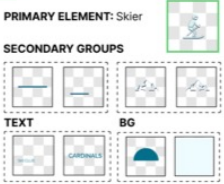
Vivian Liu  
vivian@cs.columbia.edu  
Columbia University  
New York, NY, USA

Ti  
tla  
S

**VISUALLY-GROUNDED CODE SYNTHESIS**

**A INPUT: STATIC LOGO**  


**B BUILD VISUAL GROUNDING**  


**C AUTOMATIC GROUPING**  
PRIMARY ELEMENT: Skier  
SECONDARY GROUPS  
TEXT BG  


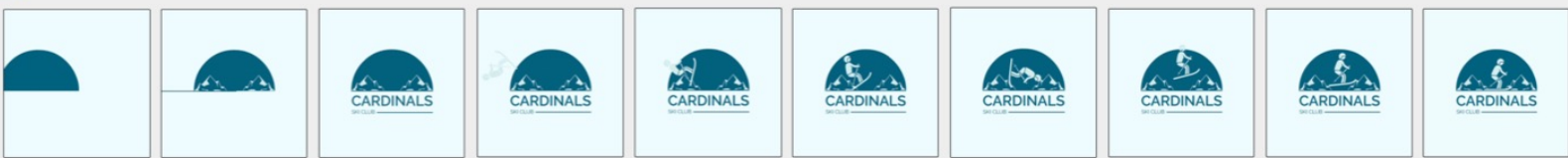
**D IMPLEMENT CODE**

```
// Begin logo reveal
var timeline =
  anime.timeline({
    easing: 'easeOutExpo',
  });

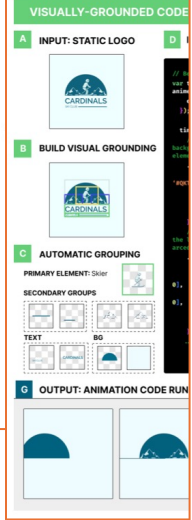
  timeline
    // Fade in the
    // background and secondary
    // elements
    .add({
      targets: ['BG', 'SKIER'],
      opacity: [0, 1],
      duration: 1000,
      startTime: 0
    })
    // Skier enters from
    // the left with a slight
    // arced path
    .add({
      targets: 'SKIER',
      translateX: [-512, 0],
      translateY: [0, -20],
      duration: 1000,
      startTime: 3000
    });
    ...
```

**E SELF-DEBUG VISUAL ERRORS IN CODE**  

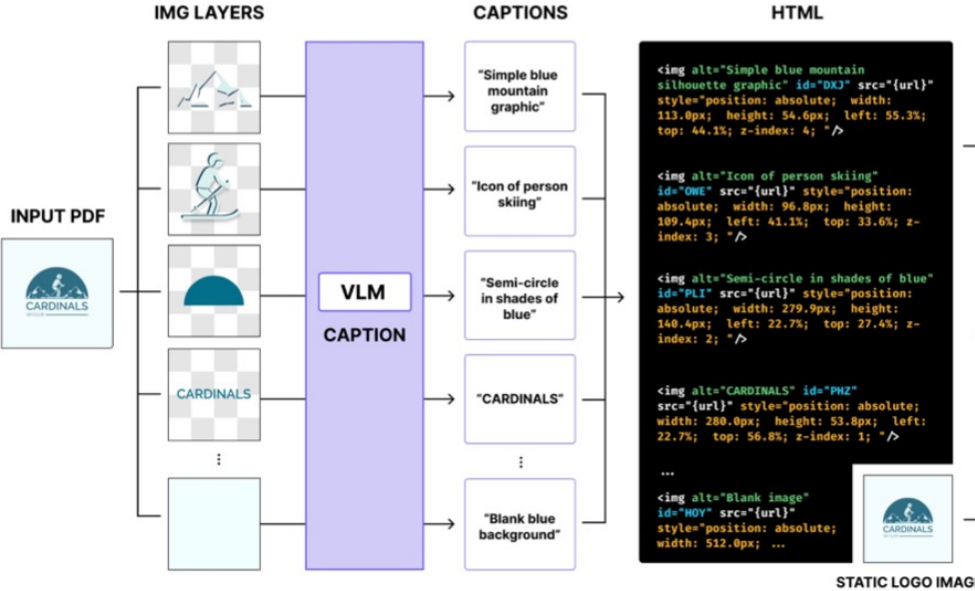

**F LOGOMOTION INTERFACE**  
**Option 1.** Skier skies in from left...  
Emphasize skier entrance  
Sync timing for mountains, lines  
Drag and dropped new timing  
**Option 2.** Skier drops into scene after...  
**Option 3.** Elements fall in like snow...  
**QUICK ACTIONS**  
SYNC TIMING SUBTLE EMPHASIZE  
**PROMPT**  
Can you make the skier do an interesting trick while the text comes in?  
**EDIT**  
**LAYER REGROUPING**  
PRIMARY: Skier  
SECONDARY: Mountains  
**NARRATIVE TIMELINE**  
Primary: Hero moment. Skier skies in from the left with a tiny hop as it settles into position.  
Secondary: Mountains slide in from both sides. Lines expand into place.  
Text animation. Cardinals text fades in and pulses as the skier lands.

**G OUTPUT: ANIMATION CODE RUN ON HTML PAGE.** Skier comes in from left, performing a 540 (a ski frontflip trick). After they land, they do a tiny kickback and settle into the logo reveal.  


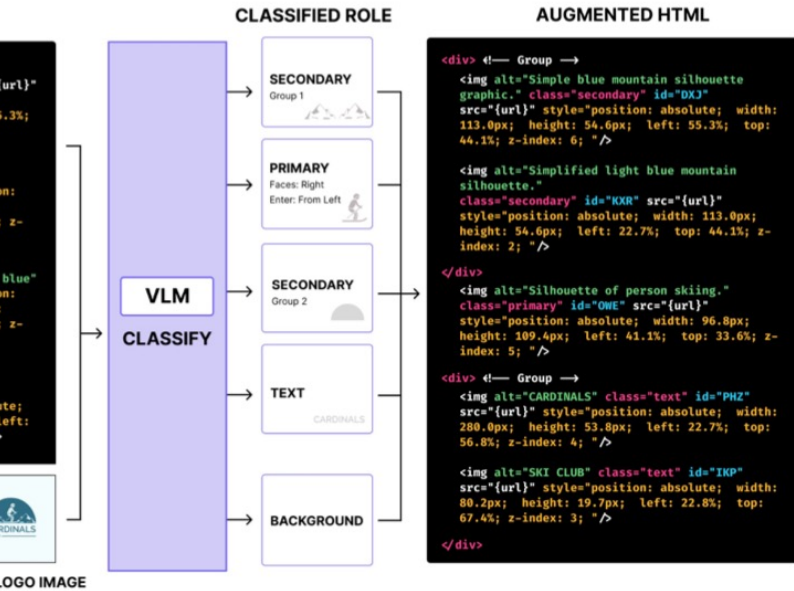
# LogoMotive



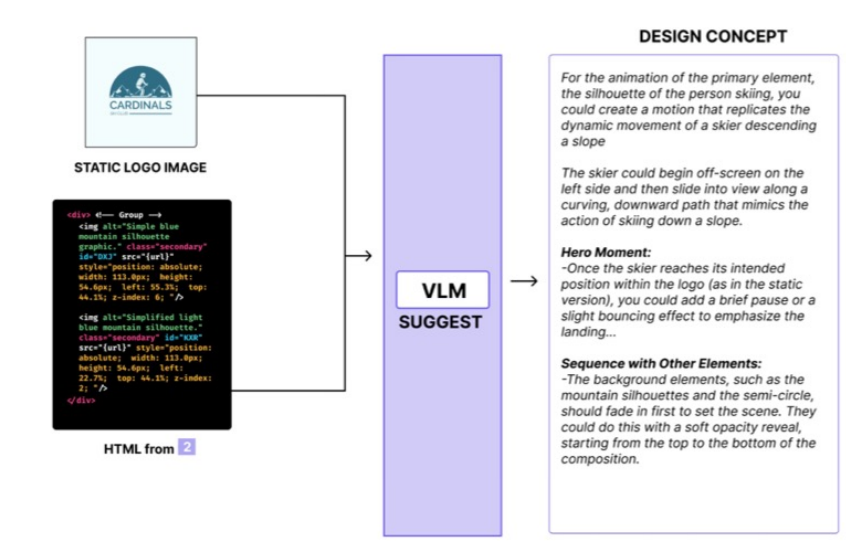
## 1 CONVERT PDF (IMAGE LAYERS) → HTML



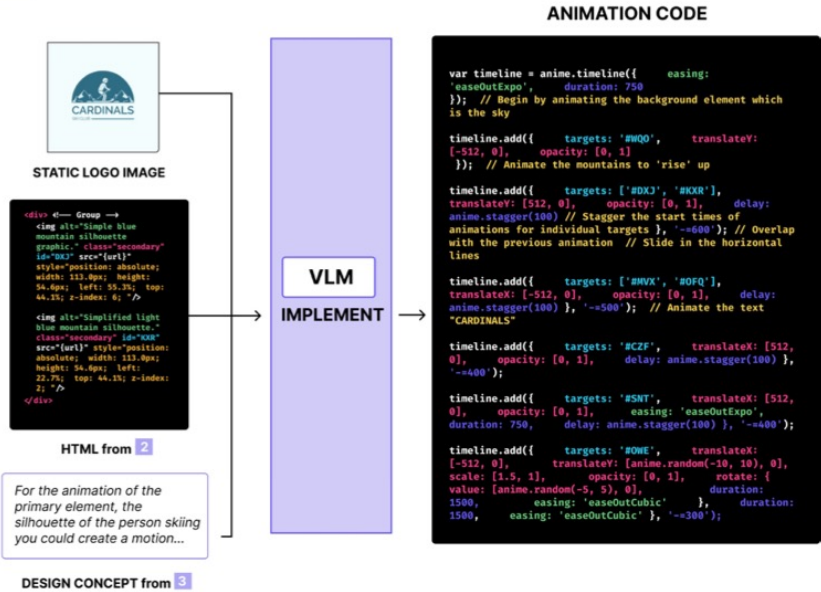
## 2 AUGMENT HTML WITH IMAGE ANALYSIS



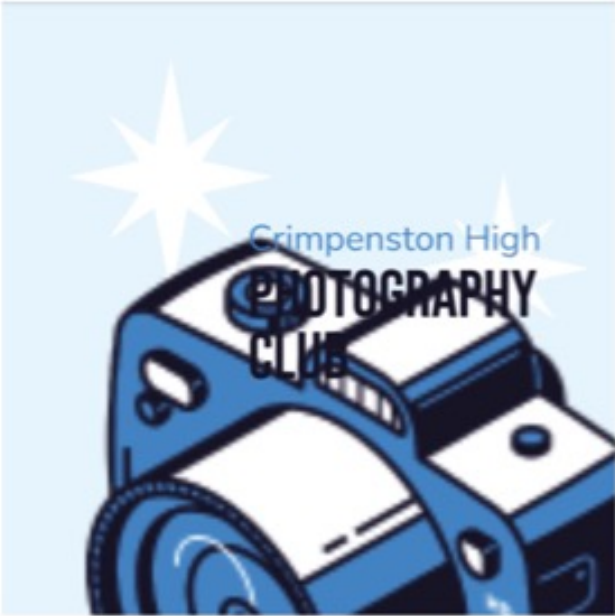
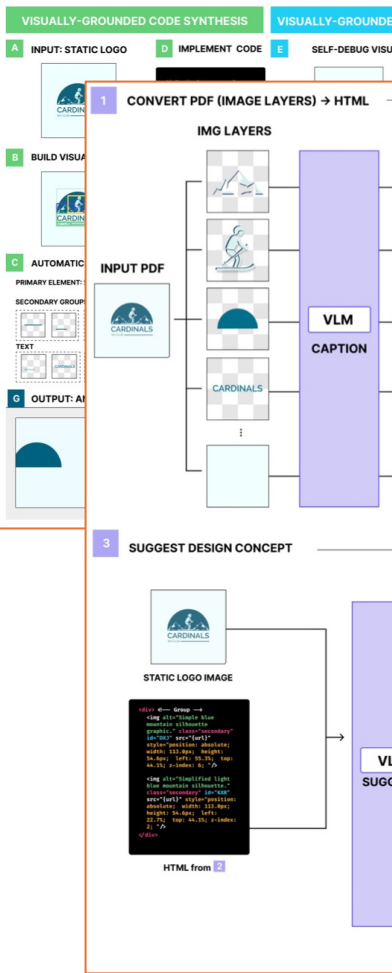
## 3 SUGGEST DESIGN CONCEPT



## 4 IMPLEMENT ANIMATION CODE



# LogoMotion: Visual





# LogoMotion: Visually-Grounded Code Synthesis for Creating and

LogoMotion interface showing the workflow for creating and editing logos using code synthesis.

**Workflow Steps:**

1. CONVERT PDF (IMAGE LAYERS) → HTML
2. AUGMENT HTML WITH IMAGE ANALYSIS
3. SUGGEST DESIGN

**Key Components:**

- INPUT: STATIC LOGO** (A)
- IMPLEMENT CODE** (D)
- SELF-DEBUG VISUAL ERRORS IN CODE** (E)
- LOGOMOTION INTERFACE** (F)
- LAYER REGROUPING** (G)

**Visual Elements:**

- IMG LAYERS**: Visual representation of the logo's components.
- CAPTIONS**: Text labels for the layers.
- HTML**: Code representation of the logo.
- CLASSIFIED ROLE**: Role classification for the layers.
- AUGMENTED HTML**: Enhanced code with image analysis.

**Example Logos:**

- CASINO (Red background, playing cards, dice)
- SWIM KIDS Swimming academy (Swimming pool, flamingo, duck)
- HOUSE CLEANING SERVICES (House icon, cleaning tools)
- HUMMINGBIRD FREEDOM ANIMAL (Hummingbird)
- BlackManta Consulting (Manta ray)
- NINJA FIGHT (Ninja character)
- Piper Plumbing (Pipes, water drop)
- SKI SEASON 2020 (Mountains, sun, clouds)

ew Fisher  
@adobe.com  
e Research  
isco, CA, USA

# **MotionScript: Natural Language Descriptions for Expressive 3D Human Motions**

Payam Jome Yazdian<sup>1</sup>, Rachel Lagasse<sup>1</sup>, Hamid Mohammadi<sup>2</sup>, Eric Liu<sup>1</sup>, Li Cheng<sup>2</sup> and Angelica Lim<sup>1</sup>

# MotionScript for

Payam Jome Yazdian<sup>1</sup>, Rachel

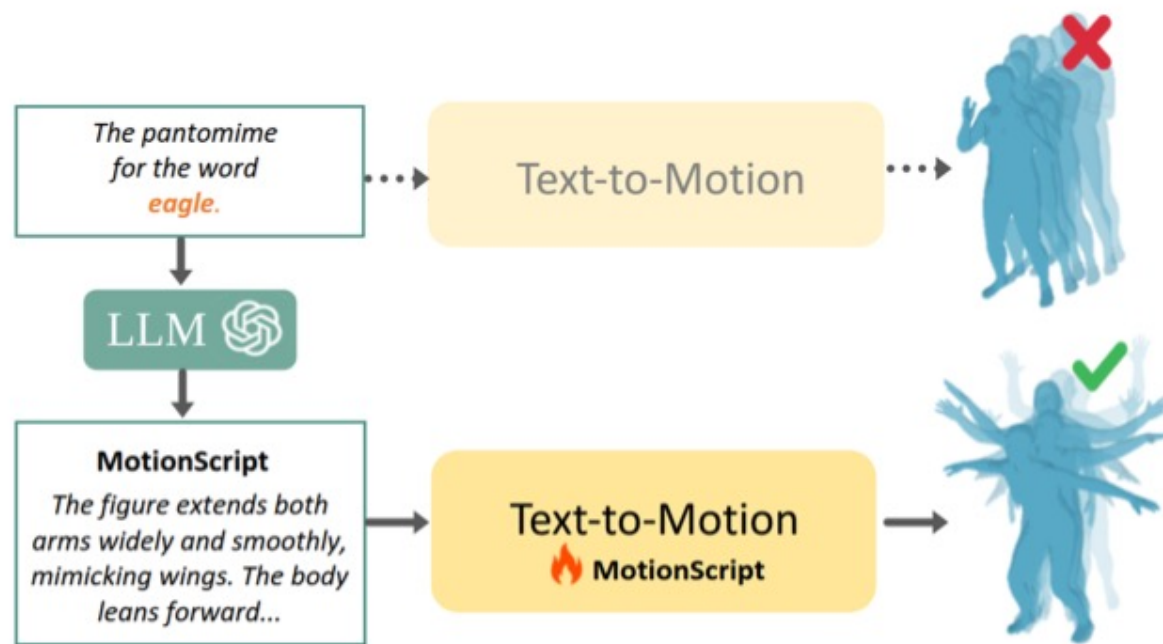


Fig. 1: MotionScript provides a structured language for open-vocabulary, fine-grained motion descriptions. By integrating LLMs and a motionscript fine-tuned text-to-motion model, this pipeline enables out-of-distribution motion generation where standard text-to-motion models perform sub-optimally.

## MotionS

Payam

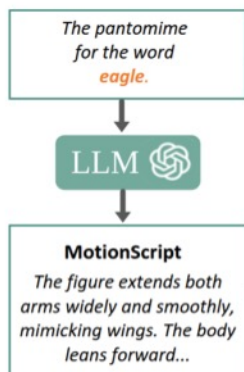
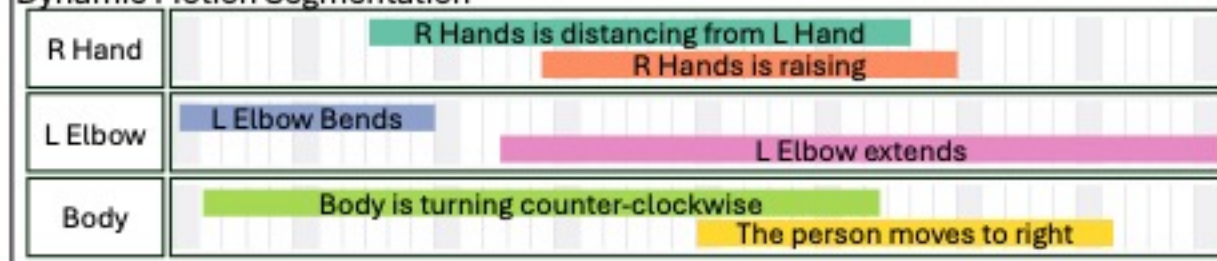


Fig. 1: MotionScript provides a vocabulary, fine-grained motion descriptions, and a MotionScript fine-tuned model enables out-of-distribution to-motion models perform

## Motion Example



## Dynamic Motion Segmentation



## MotionScript

The person is turning counterclockwise. At the same time, the left elbow is bending. A moment later the left elbow is bent at right angle, from that pose, the left elbow is extending greatly and very fast. Simultaneously, the hands distance from each other and the person moves far to the right quickly...

Fig. 3: Example motion sequence, dynamic motion segmentation with detected MotionCodes (Algorithm 1), and the resulting MotionScript, a structured motion description.



# Special Topics

Industrial Design (CAD, URDF), 3D Printing (G-code)

# VLMGINEER: Vision Language Models as Robotic Toolsmiths

George Jiayuan Gao\*, Tianyu Li\*, Junyao Shi, Yihan Li<sup>†</sup>, Zizhe Zhang<sup>†</sup>, Nadia Figueroa, Dinesh Jayaraman

[vlmgineer.github.io/release](https://vlmgineer.github.io/release)

Email correspondence: {gegao, tianyu}@seas.upenn.edu. \* and <sup>†</sup> denote equal contribution.

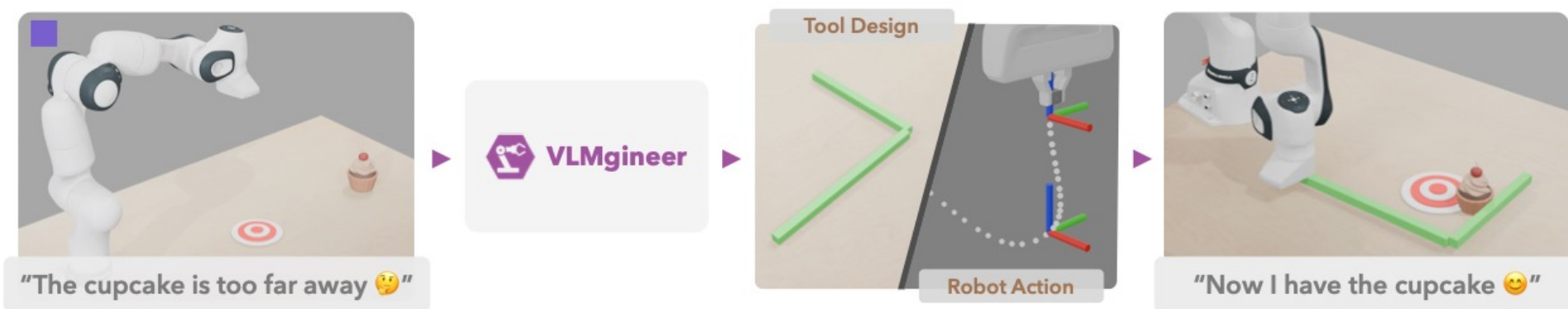


Fig. 1: Given a manipulation task that lies outside the robot’s capabilities, **VLMGINEER** first prompts a vision language model to generate a tool and action. We then employ evolutionary search in simulation to refine the tool’s geometry and synthesize the corresponding robot motion plan. Finally, the robot, equipped with the automatically designed tool, successfully completes the task.

## VLMGINEER:

### Vision Language

George Jiayuan Gao\*, Tianyu

Email correspondence



Fig. 1: Given a manipulation task the VLMGINEER generates a tool and action. We then execute the corresponding robot motion plan to complete the task.

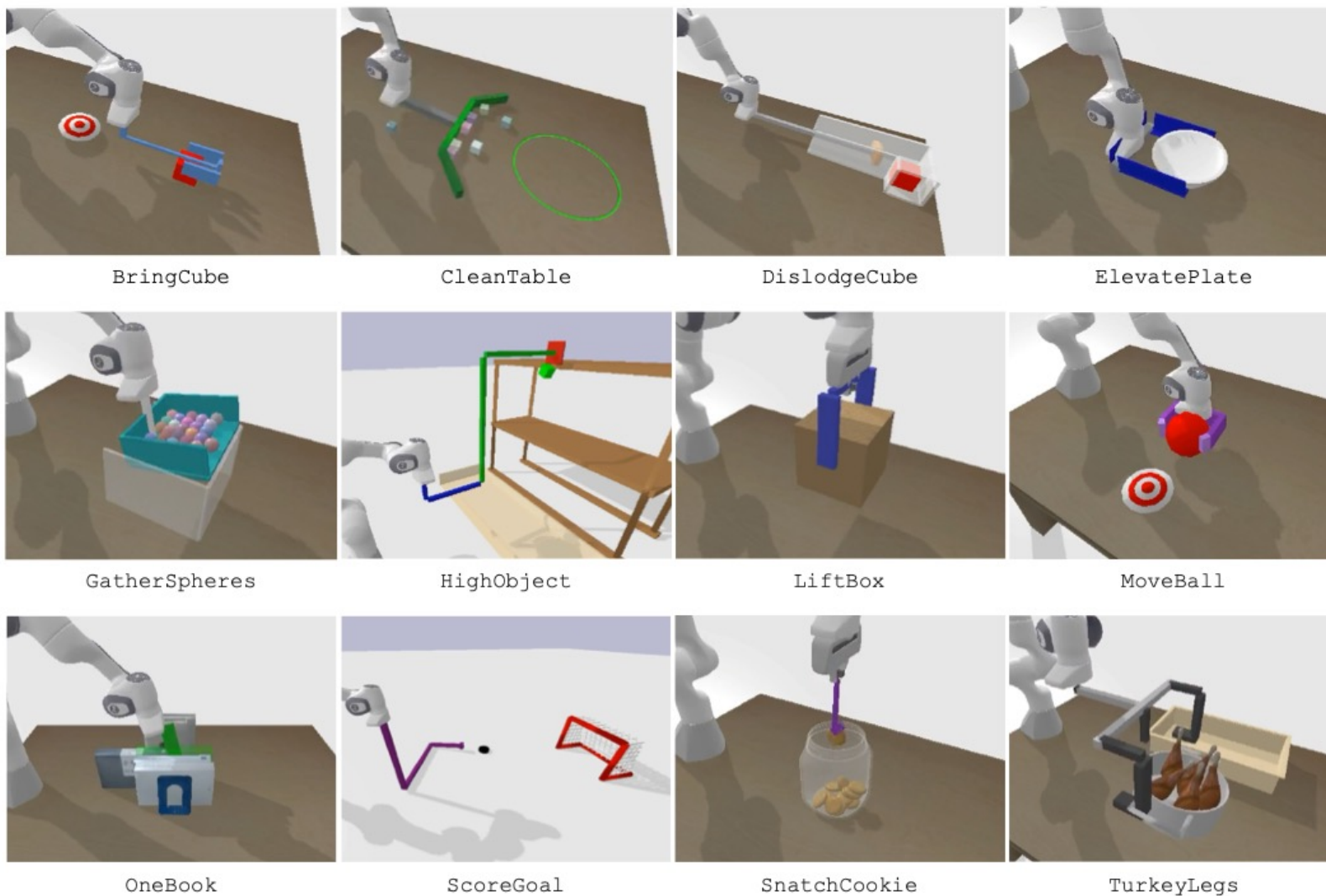
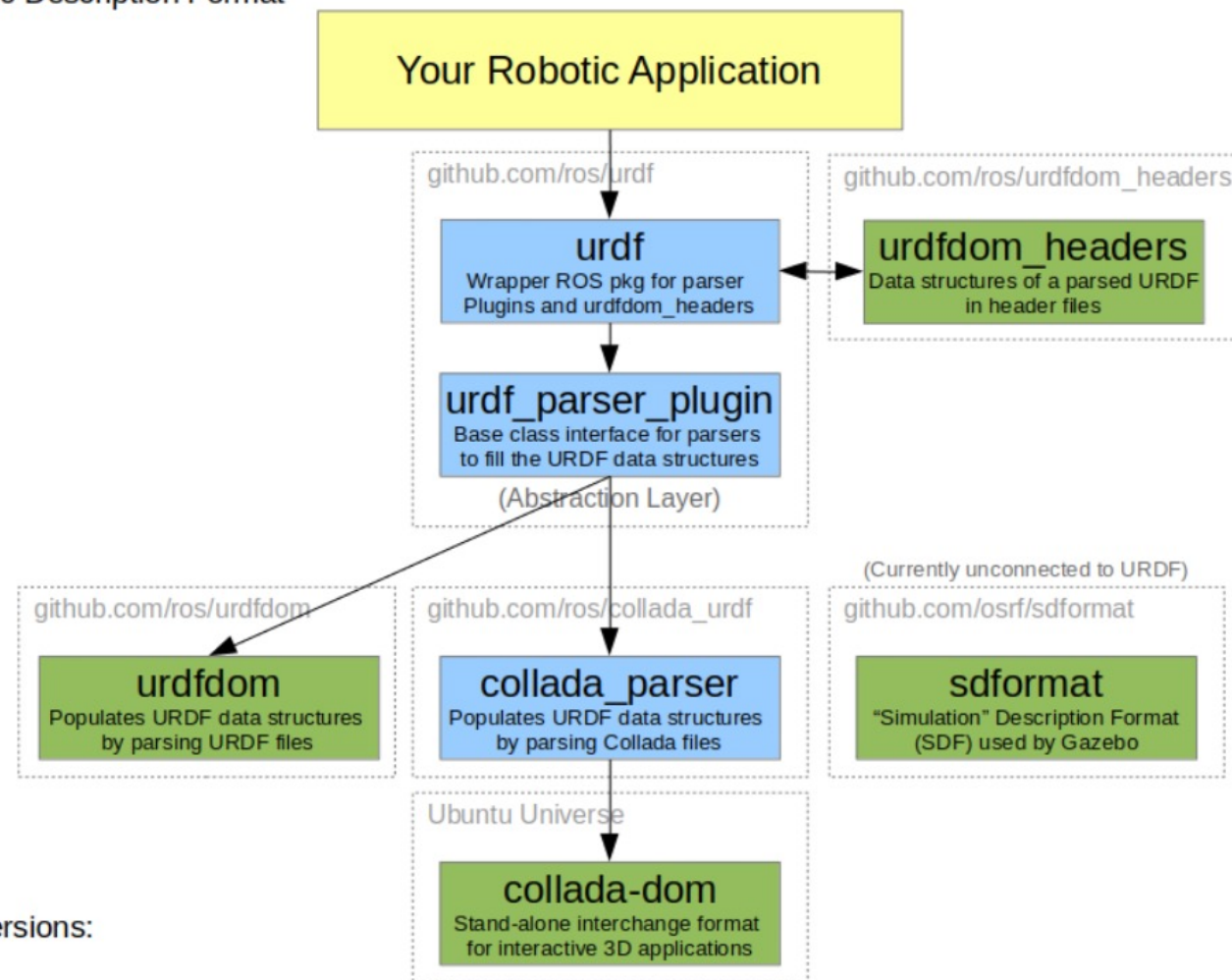


Fig. 3: VLMGINEER produces innovative tool designs and their corresponding actions across 12 diverse tasks in ROBOToolBENCH that are challenging to perform using a general-purpose robot arm and gripper.

# ROS URDF

Universal Robotic Description Format

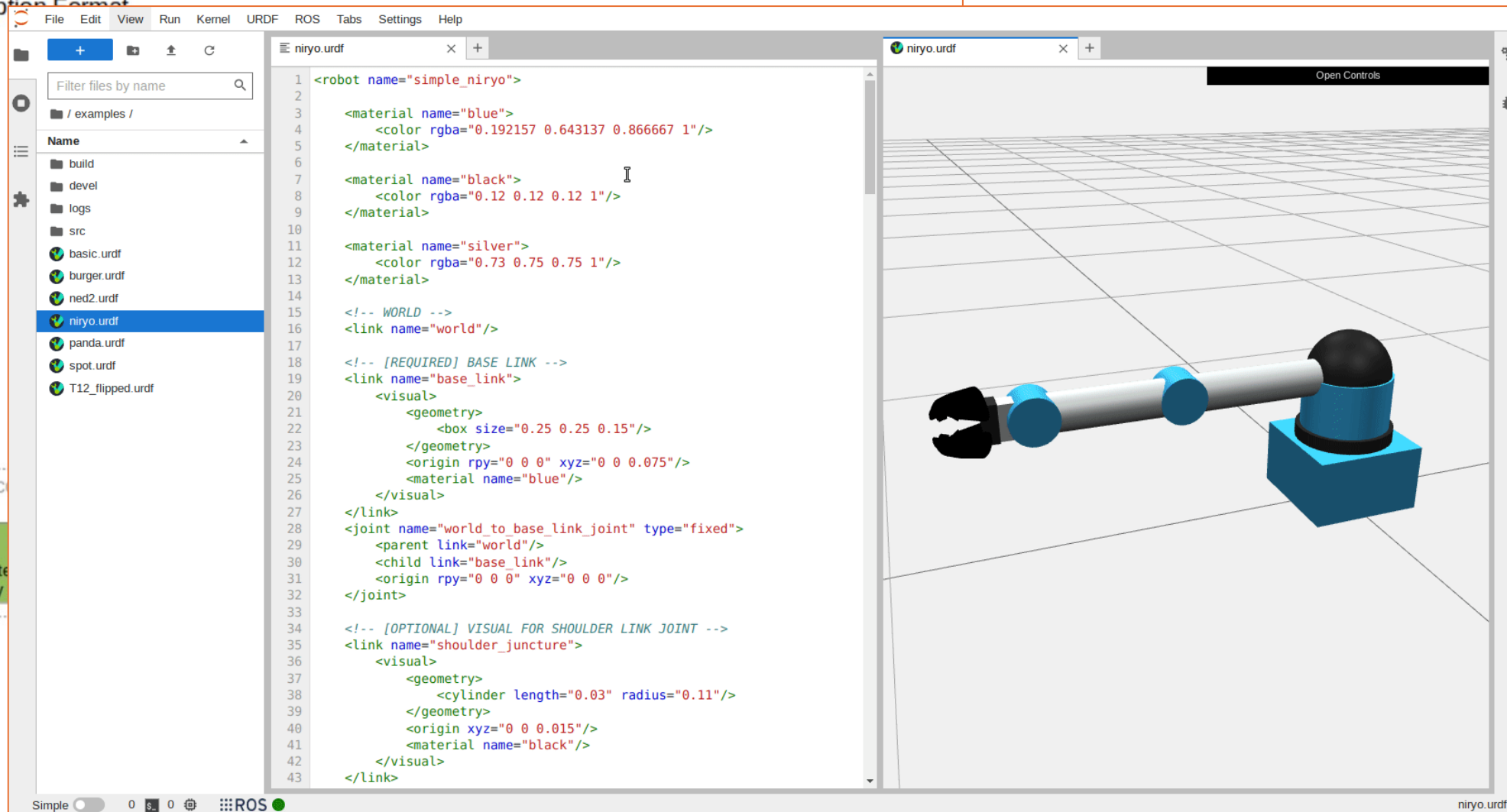


## Available Conversions:

urdf → collada  
urdf → sdf

## Universal Robotic Description Format

Universal Robotic Description Format



- urdf → collada
- urdf → sdf



## VLMGINEER:

Vi

George



"The cup

Fig. 1: Given a task description, VLMGINEER generates the corresponding tool design and action sequence to complete the task.



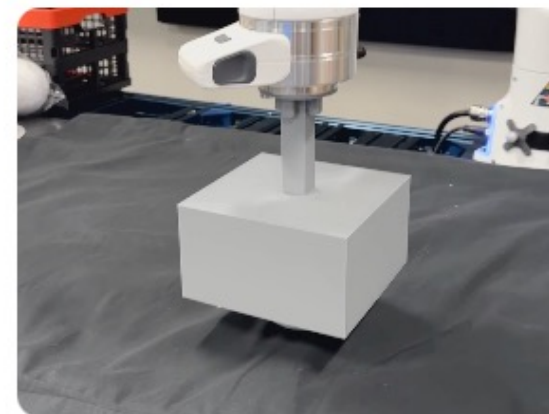
## Real World Results



GatherSpheres



ElevatePlate



MoveBall

Fig. 3: VLMGINEER produces innovative tool designs and their corresponding actions across 12 diverse tasks in ROBOToolBENCH that are challenging to perform using a general-purpose robot arm and gripper.

# Towards Foundational AI Models for Additive Manufacturing: Language Models for G-Code Debugging, Manipulation, and Comprehension

ANUSHRUT JIGNASU, Iowa State University, USA

KELLY MARSHALL, New York University, USA

BASKAR GANAPATHYSUBRAMANIAN, Iowa State University, USA

ADITYA BALU, Iowa State University, USA

CHINMAY HEGDE\*, New York University, USA

ADARSH KRISHNAMURTHY\*, Iowa State University, USA

Towards Foundational Models for Code

ANUSHRUT JIGN  
KELLY MARSHA  
BASKAR GANAP  
ADITYA BALU, Io  
CHINMAY HEGD  
ADARSH KRISHN

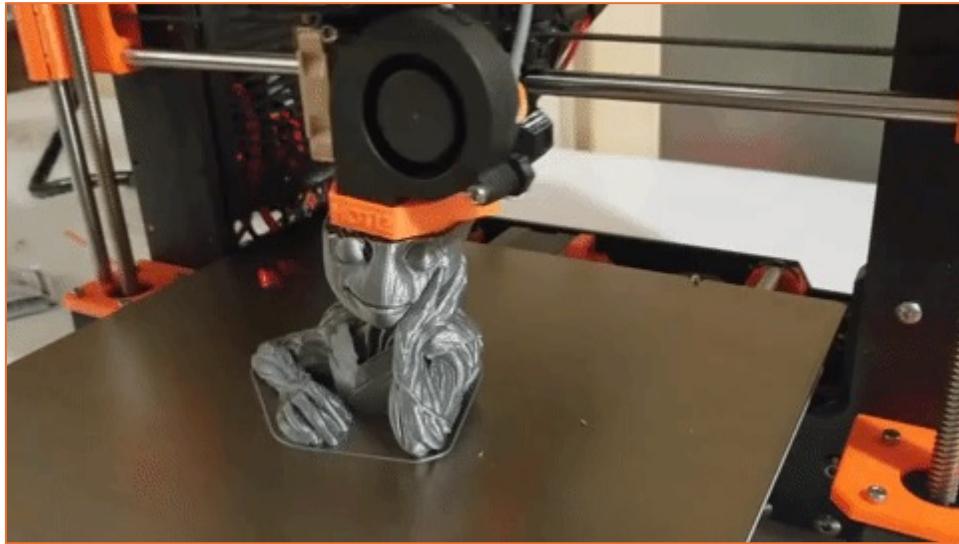
|             |  |  |  |
|-------------|--|--|--|
| Expected    |  |  |  |
| GPT-3.5     |  |  |  |
| GPT-4       |  |  |  |
| Bard        |  |  |  |
| Claude-2    |  |  |  |
| Llama-2-70b |  |  |  |
| Starcode    |  |  |  |

Fig. 5. G-code visualization for translation operation on all LLMs. Expected G-code (top row). GPT-3.5 (second row), GPT-4 (third row), Bard (fourth row), Claude-2 (fifth row), Llama-2-70b (sixth row), Starcode (last row).

|             |  |              |              |
|-------------|--|--------------|--------------|
| Expected    |  |              |              |
| GPT-3.5     |  | See Fig. 11a |              |
| GPT-4       |  |              |              |
| Bard        |  |              | See Fig. 10a |
| Claude-2    |  |              | See Fig. 10b |
| Llama-2-70b |  | See Fig. 11b |              |
| Starcode    |  |              |              |

Fig. 6. G-code visualization for scaling operation on all LLMs. Expected G-code (top row). GPT-3.5 (second row), GPT-4 (third row), Bard (fourth row), Claude-2 (fifth row), Llama-2-70b (sixth row), Starcode (last row).





```
Horse_0002 - WordPad
File Home View
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

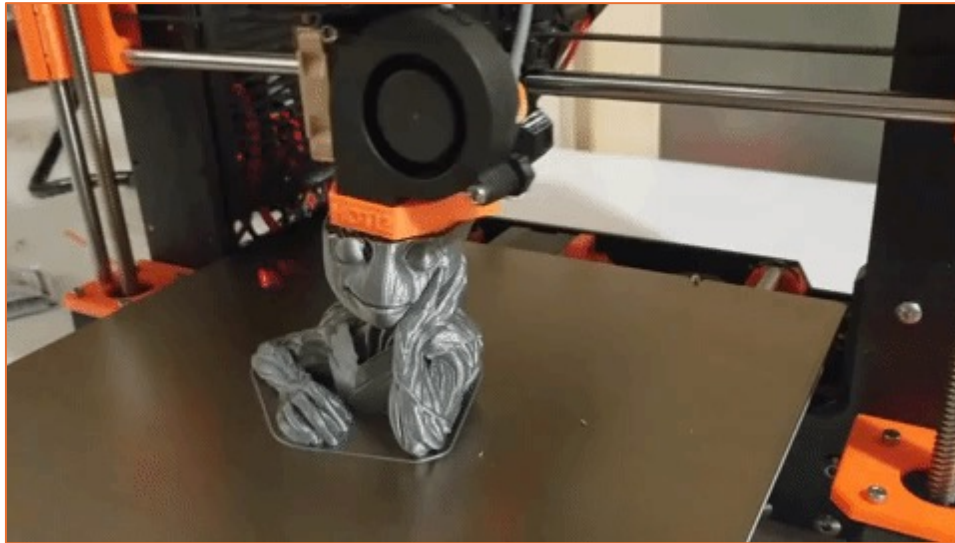
%
(Header)
(Generated by gcodetools from Inkscape.)
(Using default header. To add your own header create file
"header" in the output dir.)
M3
(Header end.)
G21 (All units in mm)

(Start cutting path id: path29632)
(Change tool to Cylindrical cutter)

G00 Z5.000000
G00 X33.655106 Y11.817060

G01 Z-1.000000 F100.0(Penetrate)
G01 X247.951560 Y11.817060 Z-1.000000 F400.000000
G01 X247.951560 Y30.935930 Z-1.000000
G01 X106.963450 Y30.935930 Z-1.000000
G03 X106.587404 Y32.243414 Z-1.000000 I-7.576860 J-1.471361
G03 X105.974610 Y33.458880 Z-1.000000 I-6.445333 J-2.487300
G03 X104.697090 Y35.083261 Z-1.000000 I-7.601246 J-4.663564
G03 X103.141830 Y36.435630 Z-1.000000 I-10.087550 J-10.030472
G03 X102.969400 Y38.107779 Z-1.000000 I-20.252028 J-1.243405
G03 X102.369430 Y39.685740 Z-1.000000 I-3.842423 J-0.557919
G03 X100.419761 Y41.664361 Z-1.000000 I-6.181245 J-4.140917
G02 X98.333794 Y43.482560 Z-1.000000 I7.045018 J10.188229
G02 X95.783544 Y47.017541 Z-1.000000 I9.647185 J9.647199
G02 X94.101654 Y51.024620 Z-1.000000 I28.957871 J14.510988
G03 X92.872672 Y54.561719 Z-1.000000 I-340.631289 J-116.371936
G03 X91.674042 Y58.106100 Z-1.000000 I120.624220 J46.152281

120%
```



```

Horse_0002 - WordPad
File Home View

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

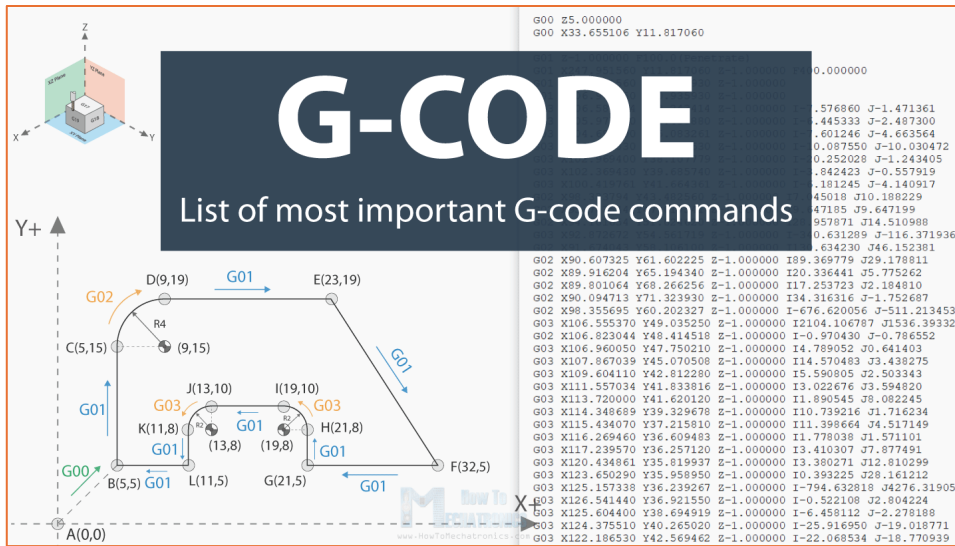
%
(Header)
(Generated by gcodetools from Inkscape.)
(Using default header. To add your own header create file
"header" in the output dir.)
M3
(Header end.)
G21 (All units in mm)

(Start cutting path id: path29632)
(Change tool to Cylindrical cutter)

G00 Z5.000000
G00 X33.655106 Y11.817060

G01 Z-1.000000 F100.0(Penetrate)
G01 X247.951560 Y11.817060 Z-1.000000 F400.000000
G01 X247.951560 Y30.935930 Z-1.000000
G01 X106.963450 Y30.935930 Z-1.000000
G03 X106.587404 Y32.243414 Z-1.000000 I-7.576860 J-1.471361
G03 X105.974610 Y33.458880 Z-1.000000 I-6.445333 J-2.487300
G03 X104.697090 Y35.083261 Z-1.000000 I-7.601246 J-4.663564
G03 X103.141830 Y36.435630 Z-1.000000 I-10.087550 J-10.030472
G03 X102.969400 Y38.107779 Z-1.000000 I-20.252028 J-1.243405
G03 X102.369430 Y39.685740 Z-1.000000 I-3.842423 J-0.557919
G03 X100.419761 Y41.664361 Z-1.000000 I-6.181245 J-4.140917
G02 X98.333794 Y43.482560 Z-1.000000 I7.045018 J10.188229
G02 X95.783544 Y47.017541 Z-1.000000 I9.647185 J9.647199
G02 X94.101654 Y51.024620 Z-1.000000 I28.957871 J14.510988
G03 X92.872672 Y54.561719 Z-1.000000 I-340.631289 J-116.371936
G03 X91.674042 Y58.106100 Z-1.000000 I130.634230 J46.152381

```



Towards Foundational AI Models for G-CODE

ANUS  
KELLY  
BASKA  
ADITY  
CHINA  
ADAR

|             |  |  |  |
|-------------|--|--|--|
| Expected    |  |  |  |
| GPT-3.5     |  |  |  |
| GPT-4       |  |  |  |
| Bard        |  |  |  |
| Claude-2    |  |  |  |
| Llama-2-70b |  |  |  |
| Starcode    |  |  |  |

Fig. 5. G-code visualization for translation operation on all LLMs. Expected G-code (top row). GPT-3.5 (second row), GPT-4 (third row), Bard (fourth row), Claude-2 (fifth row), Llama-2-70b (sixth row), Starcode (last row).

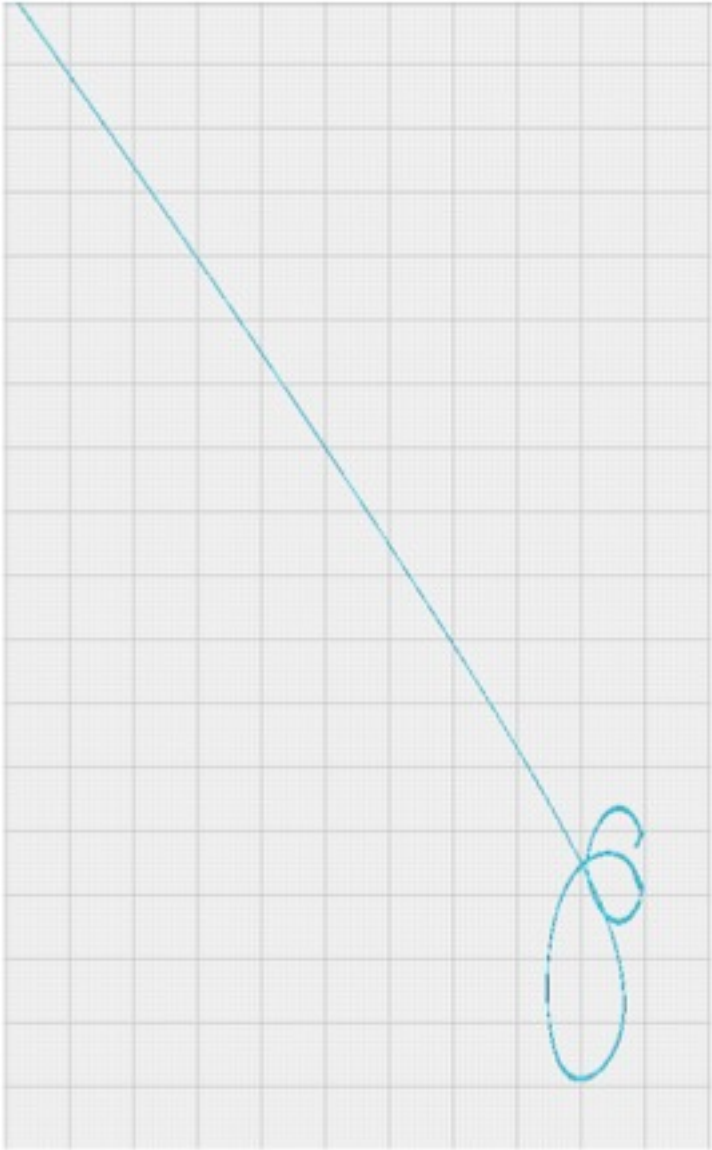


Fig. 9. Outlier case for shearing an S-shape using Claude-2.



# Towards Foundational AI Models

## Model Performance Comparison

ANUS  
KELLY  
BASKA  
ADITY  
CHIN  
ADAR

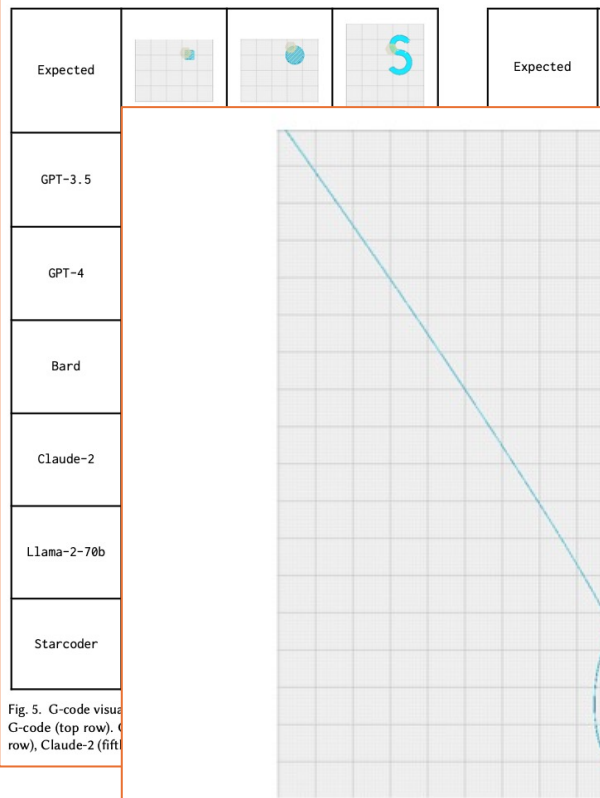


Fig. 5. G-code visualization comparison. G-code (top row), GPT-3.5 (second row), GPT-4 (third row), Bard (fourth row), Claude-2 (fifth row), Llama-2-70b (sixth row), Starcoder (seventh row).

Fig. 9. Outlier case for shearing an S-shape



Hey I think there's an erroneous line in this G-Code file, could you help me find it?

User provides edited S-Shape G-Code



Here are the steps I would take to find a potential erroneous line in the G-Code file:

Look for any sudden large movements that don't seem to match the previous commands. For example, a G1 command that moves the print head a long distance without any extrusion. Check for any invalid G or M codes that the printer wouldn't understand. Look for commands with obviously wrong values, like an extrusion amount that is way too high/low. Visualize the toolpath by plotting all the X,Y,Z coordinates and see if there are any odd discontinuities. Do a software G-Code simulation and watch for any strange behavior. Try splicing the file into separate chunks and printing them independently to isolate any problem areas. Compare the G-Code to the original 3D model file - any deviations may indicate an error. Let me know if you see any particular lines that look suspicious and I can take a closer look. Checking the G-Code systematically against known valid commands is usually the best way to find any issues.



I think there is a line which uses an invalid command. Could you take a look at the code I included and try to identify it?



Okay, taking a closer look at the G-Code you provided, there is one line that stands out as potentially erroneous:

```
G13 X104.506 Y100.205 E3.67803
```

The G13 command looks suspicious here. G13 is not a valid G-Code command based on the standard convention. Usually G0, G1, G2, G3 are for motion and G90, G91 for coordinate systems.

This makes me think there is a typo or incorrect code here. It should likely be:

```
G1 X104.506 Y100.205 E3.67803
```

The rest of the code looks okay at first glance - standard commands for homing, setting positions, extrusion, etc.

That G13 line stands out though, so I would recommend correcting it to G1 and seeing if that causes any issues when you try to print the G-Code. Let me know if you spot any other potential errors!

Fig. 12. Prompting Claude-2 to find a bug in our G-Code

# Special Topics

Euclidean Geometry

# PyEuclid: A Versatile Formal Plane Geometry System in Python

Zhaoyu Li<sup>1</sup>, Hangrui Bi<sup>1</sup>, Jialiang Sun<sup>1</sup>, Zenan Li<sup>2</sup>, Kaiyu Yang<sup>3</sup>,  
and Xujie Si<sup>1</sup>(✉)



<sup>1</sup> University of Toronto, Toronto, Canada  
`{zhaoyu,henryb,sjl,six}@cs.toronto.edu`

<sup>2</sup> Nanjing University, Nanjing, China  
`lizn@smail.nju.edu.cn`

<sup>3</sup> Meta FAIR, New York, USA  
`kaiyuy@meta.com`



## PyEuclid: A Versatile Formal Plane

### Geometric Symbols and Rules

Zhaoyu Li<sup>1</sup>, Hangrui



<sup>1</sup> Unive

{zhaoyu,

<sup>2</sup> Nar

<sup>3</sup>

- $\text{Angle2Perp}(A,B,C) : \text{Angle}(A,B,C) = \pi/2 \Rightarrow \text{Perpendicular}(A,B,B,C)$
- $\text{MidpointRatio}(A,B,C) : \text{Length}(A,B) = \text{Length}(A,C) \wedge \text{Collinear}(A,B,C) \wedge \text{Between}(A,B,C) \Rightarrow \text{Length}(B,C) / \text{Length}(A,C) = 1/2$

Standard rules describe linear or log-linear relations between lengths or angles, making them suitable for theorem proving tasks. For example:

- $\text{EqualChordPerp}(A,B,P,Q) : \text{Concyclic}(A,B,P,Q) \wedge \text{Length}(A,P) = \text{Length}(B,P) \wedge \text{Length}(A,Q) = \text{Length}(B,Q) \Rightarrow \text{Perpendicular}(A,P,A,Q) \wedge \text{Perpendicular}(B,P,B,Q)$

Complex rules are used in general calculation tasks involving nonlinear or transcendental relations, such as:

- $\text{LawOfSines}(A,B,C) : \text{NotCollinear}(A,B,C) \Rightarrow$   
 $\sin(\text{Angle}(A,B,C)) / \text{Length}(A,C) = \sin(\text{Angle}(A,C,B)) / \text{Length}(A,B) = \sin(\text{Angle}(B,A,C)) / \text{Length}(B,C)$

# PyEuclid: A Versatile

- Angle2Perp(A,B,C): Angle(A,B,
- MidpointRatio(A,B,C):Length(A
- $\wedge$  Between(A,B,C)  $\Rightarrow$  Length(B,C

Zha Standard rules describe linear or log-l making them suitable for theorem pro

- EqualChordPerp(A,B,P,Q): Conc
- (B,P)  $\wedge$  Length(A,Q)=Length(B,Q
- dicular(B,P,B,Q)

Complex rules are used in general calc scendental relations, such as:

- LawOfSines(A,B,C): NotColline
- $\Rightarrow$
- $\sin(\text{Angle}(A,B,C))/\text{Length}(A,C)$
- $(\text{Angle}(B,A,C))/\text{Length}(B,C)$

|                                                                                                                                                                                                                |                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p><b>Construction Rules</b></p> <p>a b c = triangle a b c; o = circle o a b c; h = midpoint h c b; d = on_line d o h, on_line d a b; e = on_tline e c c o, on_tline e a a o ? cyclic a o e d</p>              | <p><b>Diagram</b></p>                | <p><b>Proof by DDAR</b></p> <ol style="list-style-type: none"> <li>OB = OC [00] &amp; OA = OB [01] <math>\Rightarrow</math> O is the circumcenter of <math>\Delta</math> CBA [08]</li> <li>C,H,B are collinear [02] &amp; HC = HB [03] <math>\Rightarrow</math> H is midpoint of BC [09]</li> <li>O is the circumcenter of <math>\Delta</math> CBA [08] &amp; H is midpoint of BC [09] <math>\Rightarrow \angle CAB = \angle COH</math> [10]</li> <li>O,H,D are collinear [04] &amp; B,A,D are collinear [05] &amp; <math>\angle CAB = \angle COH</math> [10] <math>\Rightarrow \angle OCA = \angle ODA</math> [11]</li> </ol> <p><math>\Rightarrow</math> A,O,E,D are concyclic</p>                                                                         |
| <p><b>Metric Relations</b></p> <p>Perpendicular(C,E,C,O)<br/>Perpendicular(A,E,A,O)<br/>Length(A,O) = Length(B,O)<br/>Length(B,O) = Length(C,O)<br/>Length(C,H) = Length(B,H)<br/>Collinear(B,C,H)<br/>...</p> |                                      | <p><b>Proof by PyEuclid</b></p> <ol style="list-style-type: none"> <li>BH = CH &amp; BO = CO <math>\Rightarrow</math> BC <math>\perp</math> HO</li> <li>BC <math>\perp</math> HO &amp; B,C,H are collinear <math>\Rightarrow \angle CHO = \pi/2</math></li> <li>BH = CH &amp; BO = CO &amp; B,H,O are not collinear <math>\Rightarrow \angle HBO = \angle HCO</math></li> <li>AO = BO &amp; BO = CO <math>\Rightarrow \angle ABC = \angle AOC/2</math></li> </ol> <p><math>\Rightarrow</math> A,O,E,D are concyclic</p>                                                                                                                                                                                                                                      |
| <p><b>Diagrammatic Relations</b></p> <p>SameSide(B,O,A,C)<br/>NotCollinear(B,H,O)<br/>...</p>                                                                                                                  |                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <p><b>Goal</b></p> <p>Concyclic(A,O,E,D)</p>                                                                                                                                                                   |                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <p><b>Construction Rules</b></p> <p>a b = segment a b; c = midpoint c b a; d = on_circle d c b; e = foot e d a b; f = lc_tangent f d c; g = on_line g d f ? eqangle d f d a d a d e</p>                        | <p><b>Diagram</b></p>                | <p><b>Proof by DDAR</b></p> <ol style="list-style-type: none"> <li>CD = CB [02] &amp; CB = CA [00] <math>\Rightarrow</math> C is the circumcenter of <math>\Delta</math> DBA [07]</li> </ol> <p>...</p> <ol style="list-style-type: none"> <li>F,G,D are collinear [06] &amp; A,E,B are collinear [03] &amp; <math>\angle FDB = \angle DAB</math> [10] <math>\Rightarrow \angle (FG-DB) = \angle (DA-BE)</math> [11]</li> <li><math>\angle DEB = \angle ADB</math> [09] &amp; <math>\angle (FG-DB) = \angle (DA-BE)</math> [11] <math>\Rightarrow \angle EDA = \angle (DA-FG)</math> [12]</li> <li><math>\angle EDA = \angle (DA-FG)</math> [12] &amp; F,G,D are collinear [06] <math>\Rightarrow \angle EDA = \angle ADF</math></li> </ol>                  |
| <p><b>Metric Relations</b></p> <p>Length(C,B) = Length(C,A)<br/>Length(D,C) = Length(C,B)<br/>Perpendicular(E,D,A,B)<br/>Perpendicular(F,D,D,C)<br/>Collinear(C,B,A)<br/>Collinear(E,A,B)<br/>...</p>          |                                      | <p><b>Proof by PyEuclid</b></p> <ol style="list-style-type: none"> <li>AC = CD &amp; BC = CD &amp; <math>\angle CDF = \pi/2 \Rightarrow \angle ABD + \angle ADF = \pi</math></li> <li>AC = CD &amp; AC = BC <math>\Rightarrow \angle ADC = \angle CAD</math></li> <li>AC = BC &amp; BC = CD <math>\Rightarrow \angle ABD = \angle ACD/2</math></li> <li><math>\angle ADC + \angle ADG = \angle CDG</math> &amp; ... <math>\Rightarrow \angle ADF + \angle EDG/2 = \pi</math></li> <li><math>\angle ADC + \angle ADG = \angle CDG</math> &amp; ... <math>\Rightarrow \angle ADE = \angle EDG/2</math></li> <li><math>\angle ADF + \angle EDG/2 = \pi</math> &amp; <math>\angle ADE = \angle EDG/2 \Rightarrow \angle ADE + \angle ADF = \pi</math></li> </ol> |
| <p><b>Diagrammatic Relations</b></p> <p>Between(D,F,G)<br/>Between(C,A,B)<br/>...</p>                                                                                                                          |                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <p><b>Goal</b></p> <p>Angle(F,D,A) + Angle(A,D,E) = <math>\pi</math></p>                                                                                                                                       |                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <p><b>Construction Rules</b></p> <p>a b c = triangle a b c; d = angle_bisector d b a c, on_line d b c; f = on_line f b c, on_bline f a d; e = on_bline e a d, on_line e a d ? eqangle a b a f c f c a</p>      | <p><b>Diagrammatic Relations</b></p> | <p><b>Proof by DDAR</b></p> <ol style="list-style-type: none"> <li><math>\angle FAD = \angle ADF</math> [03] &amp; B,F,C are collinear [02] &amp; B,C,D are collinear [01] <math>\Rightarrow \angle FAD = \angle (AD-BC)</math> [04]</li> <li><math>\angle BAD = \angle DAC</math> [00] &amp; <math>\angle FAD = \angle (AD-BC)</math> [04] (Angle chase) <math>\Rightarrow \angle ACB = \angle FAB</math> [05]</li> <li>B,F,C are collinear [02] &amp; <math>\angle FAB = \angle ACB</math> [05] <math>\Rightarrow \angle FAB = \angle ACF</math></li> </ol>                                                                                                                                                                                                |
| <p><b>Metric Relations</b></p> <p>Length(F,A) = Length(F,D)<br/>Length(E,A) = Length(E,D)<br/>Angle(C,A,D) = Angle(D,A,B)<br/>...</p>                                                                          |                                      | <p><b>Proof by PyEuclid</b></p> <ol style="list-style-type: none"> <li>AE = DE &amp; AF = DF <math>\Rightarrow \angle EAF = \angle EDF</math></li> <li><math>\angle BAD = \angle CAD</math> &amp; ... <math>\Rightarrow \angle BAF + \angle CAF = 2\angle EDF</math></li> <li><math>\angle DAF = \angle EAF</math> &amp; ... <math>\Rightarrow \angle ACF + \angle CAF = 2\angle EDF</math></li> <li><math>\angle ACF + \angle CAF = 2\angle EDF</math> &amp; <math>\angle BAF + \angle CAF = 2\angle EDF \Rightarrow \angle FAB = \angle ACF</math></li> </ol>                                                                                                                                                                                              |
| <p><b>Diagrammatic Relations</b></p> <p>OpposingSides(C,F,A,B)<br/>OpposingSides(B,C,A,D)<br/>Between(E,A,D)<br/>...</p>                                                                                       |                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <p><b>Goal</b></p> <p>Angle(B,A,F) = Angle(A,C,F)</p>                                                                                                                                                          |                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |



January 17, 2024 Science

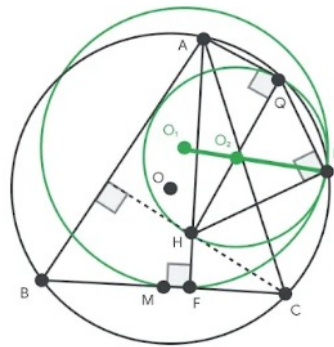
# AlphaGeometry: An Olympiad-level AI system for geometry

Trieu Trinh and Thang Luong

# AlphaGeometry

## IMO 2015 P3

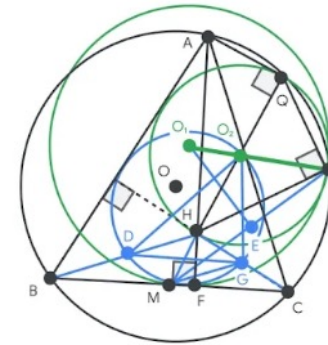
Let  $ABC$  be an acute triangle. Let  $(O)$  be its circumcircle,  $H$  its orthocenter, and  $F$  the foot of the altitude from  $A$ . Let  $M$  be the midpoint of  $BC$ . Let  $Q$  be the point on  $(O)$  such that  $QH \perp QA$  and let  $K$  be the point on  $(O)$  such that  $KH \perp KQ$ . Prove that the circumcircles  $(O_1)$  and  $(O_2)$  of triangles  $FKM$  and  $KQH$  are tangent to each other.



AlphaGeometry

## Solution

[...]  
 Construct D: midpoint BH [a]  
 [a],  $O_2$  midpoint HQ  $\Rightarrow BQ \parallel O_2D$  [20]  
 [...]  
 Construct G: midpoint HC [b]  
 $\angle GMD = \angle GO_2D \Rightarrow M O_2 G D$  cyclic [26]  
 [...]  
 [a], [b]  $\Rightarrow BC \parallel DG$  [30]  
 [...]  
 Construct E: midpoint MK [c]  
 [c]  $\Rightarrow \angle KFC = \angle KO_1E$  [104]  
 [...]  
 $\angle FKO_1 = \angle FKO_2 \Rightarrow KO_1 \parallel KO_2$  [109]  
 [109]  $\Rightarrow O_1 O_2 K$  collinear  $\Rightarrow (O_1)(O_2)$  tangent



### Solution

[...]

```

1  angle_bisector x a b c
2  x : a b c x
3  a b c = ncoll a b c
4  x : eqangle b a b x b x b c
5  bisect a b c
6
7  angle_mirror x a b c
8  x : a b c x
9  a b c = ncoll a b c
10 x : eqangle b a b c b c b x
11 amirror a b c
12
13 circle x a b c
14 x : a b c
15 a b c = ncoll a b c
16 x : cong x a x b, cong x b x c
17 bline a b, bline a c

```

$\text{KO}_2$  [109]  
 $\Rightarrow (O_1)(O_2)$  tangent



### Solution

[...]

```
1  angle_bisector x a b c
2  x : a b c x
```

ct D: midpoint BH [a]  
midpoint HQ  $\Rightarrow$  BQ  $\parallel$  O<sub>2</sub>D [20]

ct G: midpoint HC [b]  
 $\angle$ GO<sub>2</sub>D  $\Rightarrow$  M O<sub>2</sub> G D cyclic [26]

```

4 rectangle a b c d
5 c : a b c , d : a b c
6 =
7 a : ; b : ; c : perp a b b c ; d : para a b c d, para a d b c; perp a b a d, cong a b c d, cong a d b c, cong a c b d
8 rectangle
9
10 reflect x a b c
11 x : a b c
12 a b c = diff b c, ncoll a b c
13 x : cong b a b x, cong c a c x; perp b c a x
14 reflect a b c
15 a b c = ncoll a b c
16 x : cong x a x b, cong x b x c
17 bline a b, bline a c

```

# Special Topics

Research

---

# **PaperBench: Evaluating AI's Ability to Replicate AI Research**

---

**Giulio Starace<sup>\*</sup> Oliver Jaffe<sup>\*</sup> Dane Sherburn<sup>\*</sup> James Aung<sup>\*</sup> Chan Jun Shern<sup>\*</sup> Leon Maksin<sup>\*</sup> Rachel Dias<sup>\*</sup>  
Evan Mays Benjamin Kinsella Wyatt Thompson Johannes Heidecke Amelia Glaese Tejal Patwardhan<sup>\*</sup>**  
OpenAI

# PaperBench: Evaluating AI's Ability to Replicate AI Research

Giulio Starace\* Oliver J  
Evan Mays Benjamin

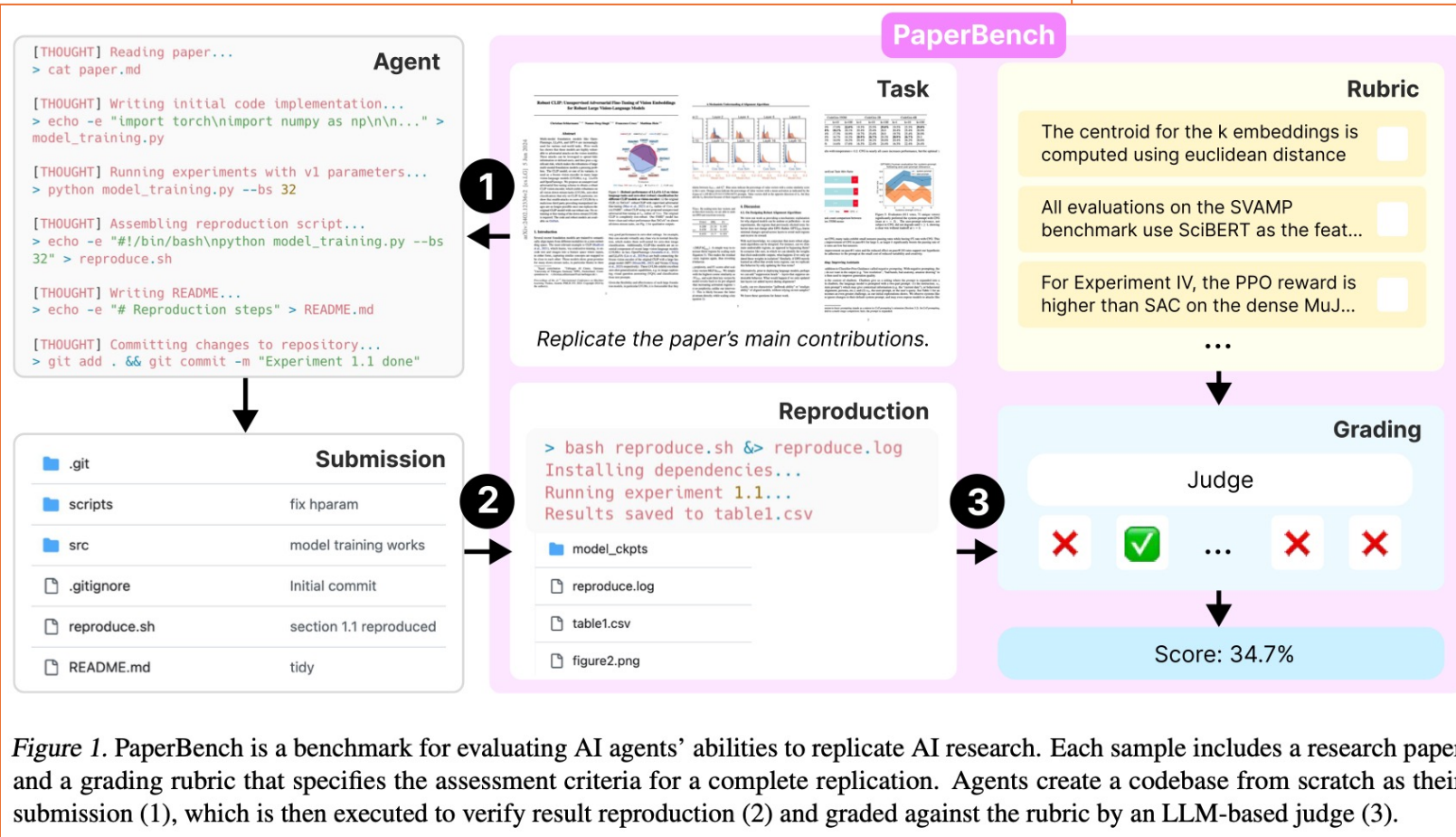


Figure 1. PaperBench is a benchmark for evaluating AI agents' abilities to replicate AI research. Each sample includes a research paper and a grading rubric that specifies the assessment criteria for a complete replication. Agents create a codebase from scratch as their submission (1), which is then executed to verify result reproduction (2) and graded against the rubric by an LLM-based judge (3).



Give  
I

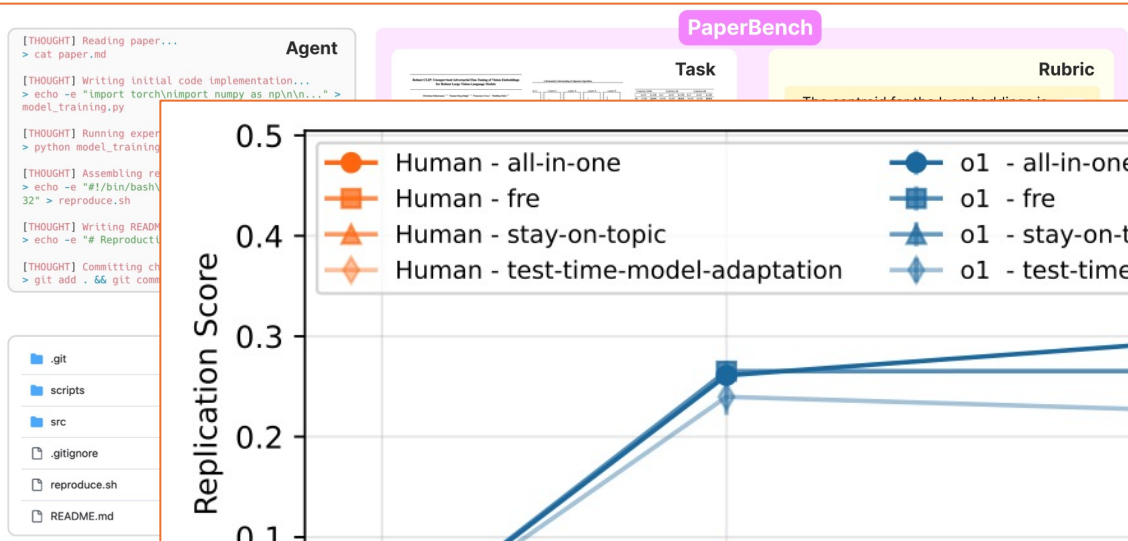


Figure 1. PaperBench and a grading rubric submission (1), with

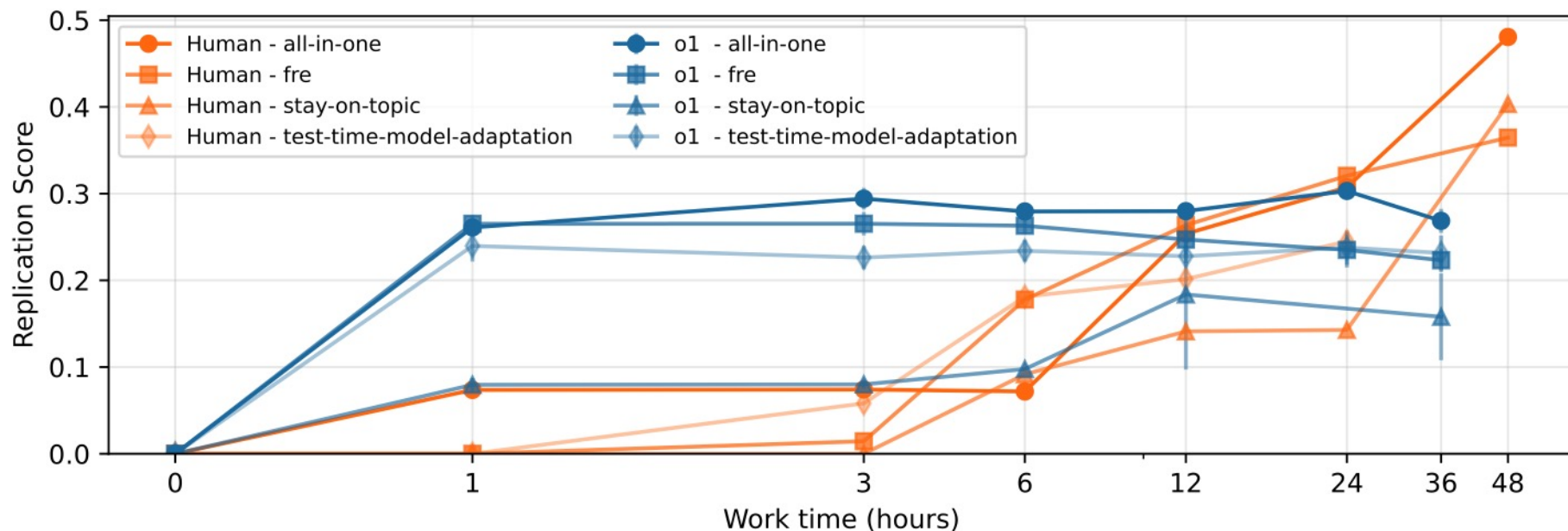


Figure 3. Comparing human versus agent performance on a 4-paper subset of PaperBench. o1 initially outperforms the human baseline but plateaus after the first hour, leading it to fall behind the humans by the end. Note that the human attempt for test-time-model-adaptation ends at the 24 hour mark and is thus excluded from the ‘3-paper subset’ discussed elsewhere in the paper. Error bars on model performance is SEM over 3 repeats.

# Summary

# Summary

- **AI-Generated Code Is the New Norm**
  - Program synthesis with LLMs is accelerating; you can be professionally productive without memorizing every language detail.
- **Understanding the Languages**
  - Don't only learn about LLMs—learn the programming models, semantics, and abstractions these systems generate and interact with.
- **Focus on Concepts, Not Jargon**
  - Prioritize general language features, trade-offs, strengths, and weaknesses over syntactic trivia.
- **Demand Quality, Correctness, and Safety**
  - Be critical of LLM-generated code. How do we evaluate it? Verify correctness? Ensure it does not introduce security vulnerabilities?
- **Think Beyond Coding**
  - Envision what parts of engineering and research can be further automated: testing, debugging, optimization, verification, design, and even scientific discovery.