

Hands Off the Terminal: LLM Agents that Build, Test, Analyze, and Reproduce

Michael Pradel (CISPA, Germany)

Joint work with Islem Bouzenia, Doehyun Baek, and Cristian Cadar



Motivation

Your research idea

Apply

Arbitrary project

Motivation



Prerequisite for many SE experiments:

Project is set up and tests are ready-to-run

Motivation



⋮

Setting up an arbitrary project is **hard**

- Many languages, build tools, and test tools
- Complex dependencies
- Informal, inconsistent, and missing documentation

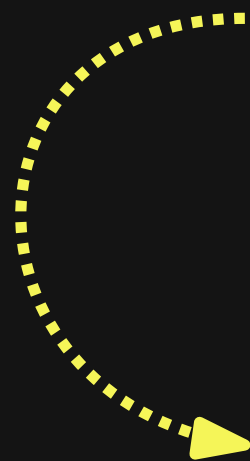
Motivation



⋮

Setting up an arbitrary project is hard

- Many languages, build tools, and test tools
- Complex dependencies
- Informal, inconsistent, and missing documentation



E.g., automated scripts of GitBugs-Java:

66,042 repositories → 228 executable test suites

More Motivation

E.g., static analyzers,
fuzzers, profilers

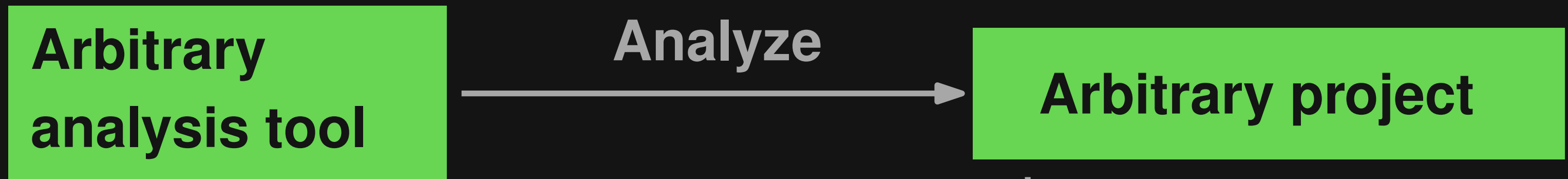
**Arbitrary
analysis tool**

Analyze

Arbitrary project

More Motivation

E.g., static analyzers,
fuzzers, profilers



Applying analysis tools is hard

- Setup of analysis tool
- Setup of target project
- Applying analysis in a **meaningful** way

Even More Motivation

A Very Promising Research Paper

Author One
Great Institution
Country A

Author Two
Great Institution
Country A

Abstract
As software is evolving, code changes can introduce regression bugs or affect the behavior in other unintended ways. Traditional regression test generation is impractical for detecting unintended behavioral changes, because it reports all behavioral differences as potential regressions. However, most code changes are intended to change the behavior in some way, e.g., to fix a bug or to add a new

PR title:
ENH: stats.differential_entropy: add array API support

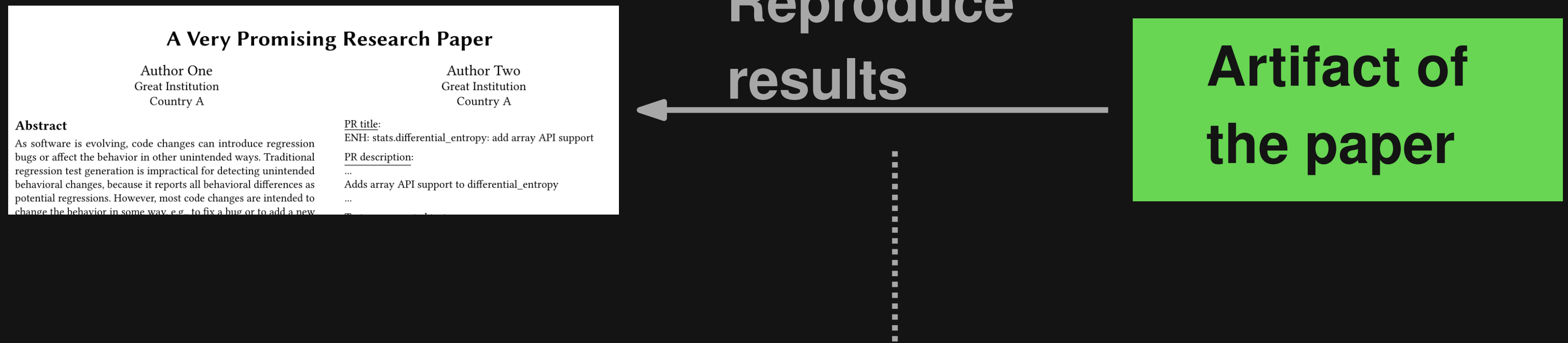
PR description:
...
Adds array API support to differential_entropy
...

Reproduce
results



**Artifact of
the paper**

Even More Motivation



Reproducing research results is hard

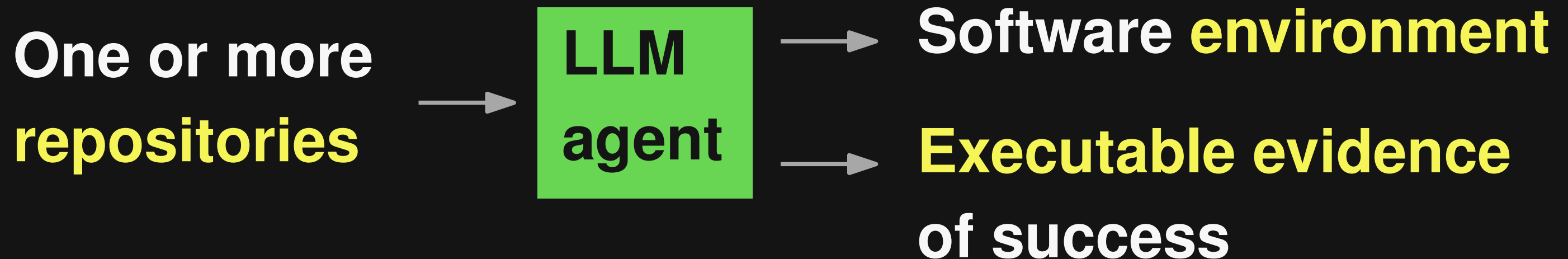
- Limited quality of research prototypes
- Outdated dependencies
- Implicit assumptions

What Do These Tasks Have in Common?

- Heavy interaction with the **command line**
- Lots of **trial and error**
- Conceptually easy, but **tedious in practice**

LLM Agents to the Rescue

Task (e.g., build, test,
analyze, or reproduce)



LLM Agents to the Rescue

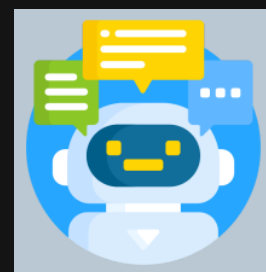
Task (e.g., build, test, analyze, or reproduce)

One or more
repositories

**LLM
agent**

Software environment

**Executable evidence
of success**



LLM

Autonomous
feedback loop



Tools

This Talk

1) **ExecutionAgent** [Bouzenia & Pradel, ISSTA'25]

- You Name It, I Run It:

An LLM Agent to Execute Tests of Arbitrary Projects

2) **AnalysisAgent** [Bouzenia, Pradel, Cadar (arXiv'26)]

- Evaluating LLM Agents on Automated Software Analysis Tasks

3) **Artisan** [Baek, Pradel (arXiv'26)]

- Artisan: Agentic Artifact Evaluation

Why Not Use a General-Purpose Agent?

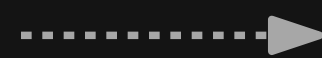
Observations

- Agents **get lost** when performing complex tasks
- A few **hard subtasks** cause many failures
- LLMs like taking **short-cuts** and **declare success prematurely**

Why Not Use a General-Purpose Agent?

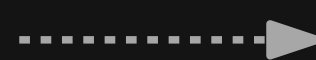
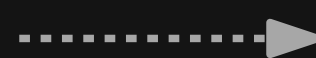
Observations

- Agents **get lost** when performing complex tasks
- A few **hard subtasks** cause many failures
- LLMs like taking **short-cuts** and **declare success prematurely**



Mitigations

- Define and enforce **task-specific stages**
- Provide **specialized tools** to overcome key obstacles
- Systematically **validate (intermediate) results**



ExecutionAgent: Task

URL of git repository

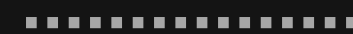


ExecutionAgent



Scripts to

- **set up project** in isolated environment
- **execute test suite** of project



- **Fully automated**
- **Technology-agnostic**

ExecutionAgent: Overview

Preparation phase:

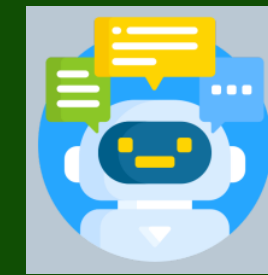
Meta-prompting

Web search

↓
Guidelines for languages, containerization techniques, and CI scripts

↓
Installation hints

Feedback phase:



LLM

Autonomous feedback loop



Tools

Prompt ingredients

Meta-Prompting

Example:

I want to install and run tests of a Java project. Give a list of 10 guidelines that I should follow to achieve my task (e.g., to figure out installation steps, dependencies, how to run tests) while considering latest practices and technologies.

Meta-Prompting

Response:

1. Review Project Documentation and README

Why: Documentation typically includes installation steps, dependencies, and setup instructions.

What to do: Check the README.md or other documentation files for prerequisites and specific instructions for installation and testing.

2. Verify Java Version Compatibility (...)

3. Check for Dependency Management Tool (...)

(...)

ExecutionAgent: Tools

- **Wrapped terminal**

- Liveness check after configurable timeout
- Remove terminal colors and special characters

- **Read files**

- **Write files**

- **Declare success**

- LLM must provide nb. of passed, failed, and skipped tests

ExecutionAgent: Example

Dockerfile

```
FROM ubuntu:20.04
# Set the timezone to avoid interactive prompts
ENV TZ=XXX/YYY # omitted for anonymity
RUN ln -snf /usr/share/zoneinfo/$TZ /etc/localtime && echo $TZ
# Update package lists and install necessary build tools and de
RUN apt-get update && \
apt-get install -y --no-install-recommends \
sudo \
build-essential \
liblzo2-dev \
libpam0g-dev \
liblz4-dev \
libcap-ng-dev \
libnl-genl-3-dev \
linux-libc-dev \
autoconf \
automake \
libtool \
cmocka-dev \
python3-docutils \
git \
&& apt-get clean && rm -rf /var/lib/apt/lists/* || exit 0
WORKDIR /app
# Clone the OpenVPN repository
RUN git clone https://github.com/OpenVPN/openvpn.git || exit 0
WORKDIR /app/openvpn
CMD ["bash"]
```

install_and_run.sh

```
#!/bin/bash
# Run autoconf to prepare the build system
autoreconf -fvi
# Configure the build system
./configure --enable-werror
# Compile the project
make -j$(nproc)
# Run tests
make check VERBOSE=1
echo "Setup_and_testing_complete."
```

Test results

```
Total: 91
Pass: 85
Fail: 4
Skip: 2
```

ExecutionAgent: Results

50 projects implemented in **14 languages**; GPT-4o-mini

	Execution- Agent	LLM scripts	AutoGPT	Flapy
Built/installed	41 / 50	29 / 50	9 / 50	6 / 10
Executed tests	33 / 50	5 / 50	4 / 50	0 / 10
Results close to ground truth *	29 / 50	4 / 50	2 / 50	0 / 10

* Up to 10% deviation from ground truth numbers of passing/failing/skipped tests

This Talk

1) **ExecutionAgent** [Bouzenia & Pradel, ISSTA'25]

- You Name It, I Run It:

An LLM Agent to Execute Tests of Arbitrary Projects

2) **AnalysisAgent** [Bouzenia, Pradel, Cadar (arXiv'26)]

- Evaluating LLM Agents on Automated Software Analysis Tasks

3) **Artisan** [Baek, Pradel (arXiv'26)]

- Artisan: Agentic Artifact Evaluation

This Talk

1) **ExecutionAgent** [Bouzenia & Pradel, ISSTA'25]

- You Name It, I Run It:

An LLM Agent to Execute Tests of Arbitrary Projects

2) **AnalysisAgent** [Bouzenia, Pradel, Cadar (arXiv'26)]

- Evaluating LLM Agents on Automated Software Analysis Tasks

3) **Artisan** [Baek, Pradel (arXiv'26)]

- Artisan: Agentic Artifact Evaluation

AnalysisAgent: Task

URL of analysis tool

(E.g., AFL++,
KLEE, Infer)

URL of target project

(Popular C, C++,
Java projects)

AnalysisAgent

```
graph LR; A[URL of analysis tool] --> B[AnalysisAgent]; C[URL of target project] --> B; B --> D[Docker container]; B --> E[Trace of executed commands]; B --> F[Tool-specific analysis results];
```

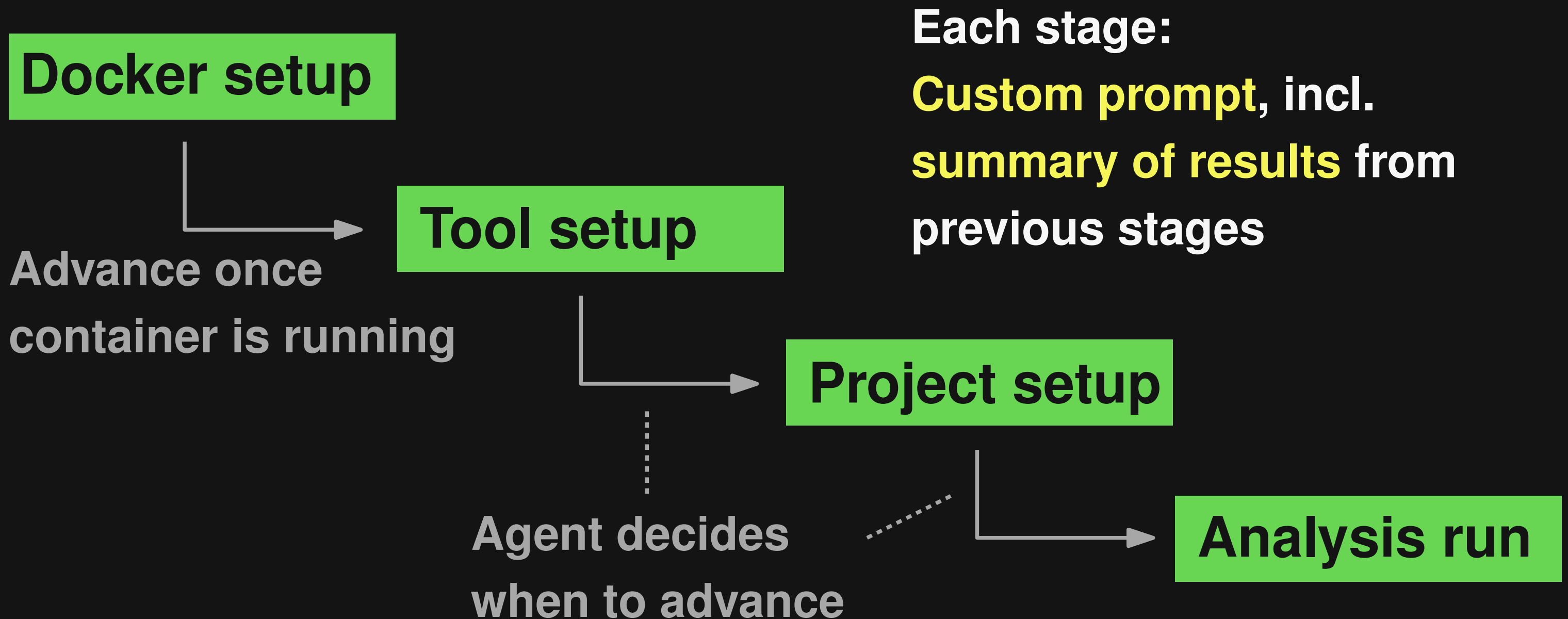
Docker container

Trace of executed commands

Tool-specific analysis results

(E.g., warnings, test cases, call graphs)

AnalysisAgent: Overview



AnalysisAgent: Validation of Results

Failure pattern of baseline agents: Premature termination

- Analysis tool is executed with `--help` only
- Analysis of toy examples
- Misconfigured analysis produces no results

AnalysisAgent: Validation of Results

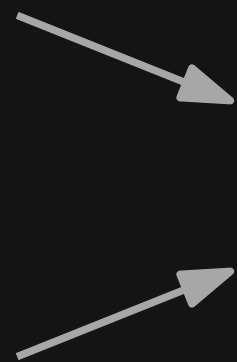
Failure pattern of baseline agents: Premature termination

- Analysis tool is executed with `--help` only
- Analysis of toy examples
- Misconfigured analysis produces no results

Stage

summaries

Analysis
outputs



LLM-as-judge

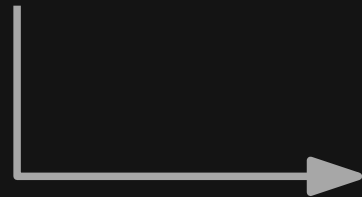
Valid output

OR

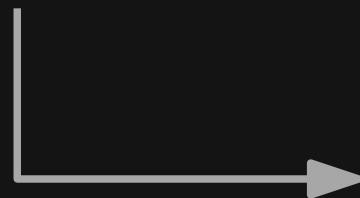
**Reject → continue
agentic loop**

Example: Apply KLEE to Fastfetch

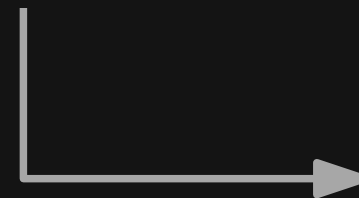
Docker setup



Tool setup



Project setup



Analysis run

Example: Apply KLEE to Fastfetch

Docker setup

Tool setup

Project setup

Analysis run

Ubuntu 22.04 with a minimal build toolchain (Git, CMake, Clang, and related utilities)

Example: Apply KLEE to Fastfetch

Docker setup

Tool setup

Installs LLVM-14, Z3, and other dependencies, then KLEE itself.

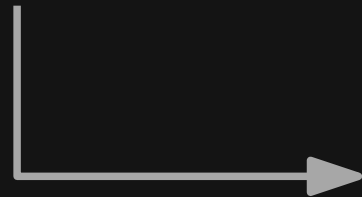
Checks `klee --version`

Project setup

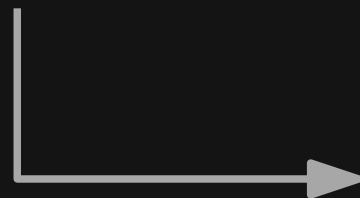
Analysis run

Example: Apply KLEE to Fastfetch

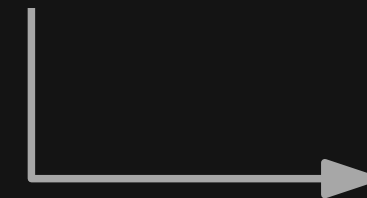
Docker setup



Tool setup



Project setup



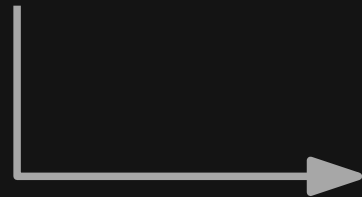
Analysis run

As KLEE needs bitcode, builds project with `wl1vm` for whole-program bitcode instrumentation and runs `extract-bc`.

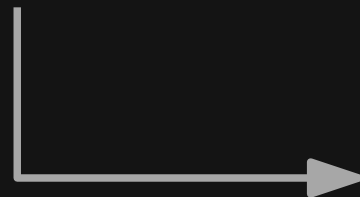
Checks for `fastfetch.bc`

Example: Apply KLEE to Fastfetch

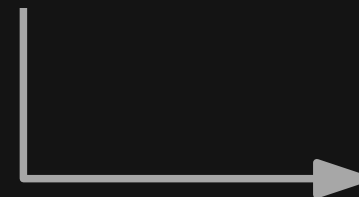
Docker setup



Tool setup



Project setup



Analysis run

Runs KLEE on `fastfetch.bc` with conservative resource limits. Produces concrete test cases.

AnalysisAgent: Results

7 analysis tools, 10 target projects

Agent	GPT-5-nano		GPT-5-mini		DeepSeek-V3.2		Gemini-3-Flash		Avg. Verif.
	Self-val.	Verif.	Self-val.	Verif.	Self-val.	Verif.	Self-val.	Verif.	
RAG-Agent	73%	9%	98%	6%	49%	3%	93%	23%	10%
Mini-SWE-Agent	100%	9%	100%	20%	100%	57%	87%	63%	37%
ExecutionAgent	30%	40%	63%	54%	50%	57%	86%	77%	57%
AnalysisAgent	77%	54%	97%	75%	100%	94%	100%	94%	79%

AnalysisAgent: Results

7 analysis tools, 10 target projects

Agent	GPT-5-nano		GPT-5-mini		DeepSeek-V3.2		Gemini-3-Flash		Avg. Verif.
	Self-val.	Verif.	Self-val.	Verif.	Self-val.	Verif.	Self-val.	Verif.	
RAG-Agent	73%	9%	98%	6%	49%	3%	93%	23%	10%
Mini-SWE-Agent	100%	9%	100%	20%	100%	57%	87%	63%	37%
ExecutionAgent	30%	40%	63%	54%	50%	57%	86%	77%	57%
AnalysisAgent	77%	54%	97%	75%	100%	94%	100%	94%	79%

Strong models \implies Higher success rate

AnalysisAgent: Results

7 analysis tools, 10 target projects

Agent	GPT-5-nano		GPT-5-mini		DeepSeek-V3.2		Gemini-3-Flash		Avg. Verif.
	Self-val.	Verif.	Self-val.	Verif.	Self-val.	Verif.	Self-val.	Verif.	
RAG-Agent	73%	9%	98%	6%	49%	3%	93%	23%	10%
Mini-SWE-Agent	100%	9%	100%	20%	100%	57%	87%	63%	37%
ExecutionAgent	30%	40%	63%	54%	50%	57%	86%	77%	57%
AnalysisAgent	77%	54%	97%	75%	100%	94%	100%	94%	79%



Huge gap between self-declared and actual success

- Especially for weaker models and baseline agents

AnalysisAgent: Results

7 analysis tools, 10 target projects

Agent	GPT-5-nano		GPT-5-mini		DeepSeek-V3.2		Gemini-3-Flash		Avg. Verif.
	Self-val.	Verif.	Self-val.	Verif.	Self-val.	Verif.	Self-val.	Verif.	
RAG-Agent	73%	9%	98%	6%	49%	3%	93%	23%	10%
Mini-SWE-Agent	100%	9%	100%	20%	100%	57%	87%	63%	37%
ExecutionAgent	30%	40%	63%	54%	50%	57%	86%	77%	57%
AnalysisAgent	77%	54%	97%	75%	100%	94%	100%	94%	79%

AnalysisAgent outperforms general-purpose agents

This Talk

1) **ExecutionAgent** [Bouzenia & Pradel, ISSTA'25]

- You Name It, I Run It:

An LLM Agent to Execute Tests of Arbitrary Projects

2) **AnalysisAgent** [Bouzenia, Pradel, Cadar (arXiv'26)]

- Evaluating LLM Agents on Automated Software Analysis Tasks

3) **Artisan** [Baek, Pradel (arXiv'26)]

- Artisan: Agentic Artifact Evaluation

This Talk

1) **ExecutionAgent** [Bouzenia & Pradel, ISSTA'25]

- You Name It, I Run It:

An LLM Agent to Execute Tests of Arbitrary Projects

2) **AnalysisAgent** [Bouzenia, Pradel, Cadar (arXiv'26)]

- Evaluating LLM Agents on Automated Software Analysis Tasks

3) **Artisan** [Baek, Pradel (arXiv'26)]

- Artisan: Agentic Artifact Evaluation

Artisan: Task

Paper

Artifact URL

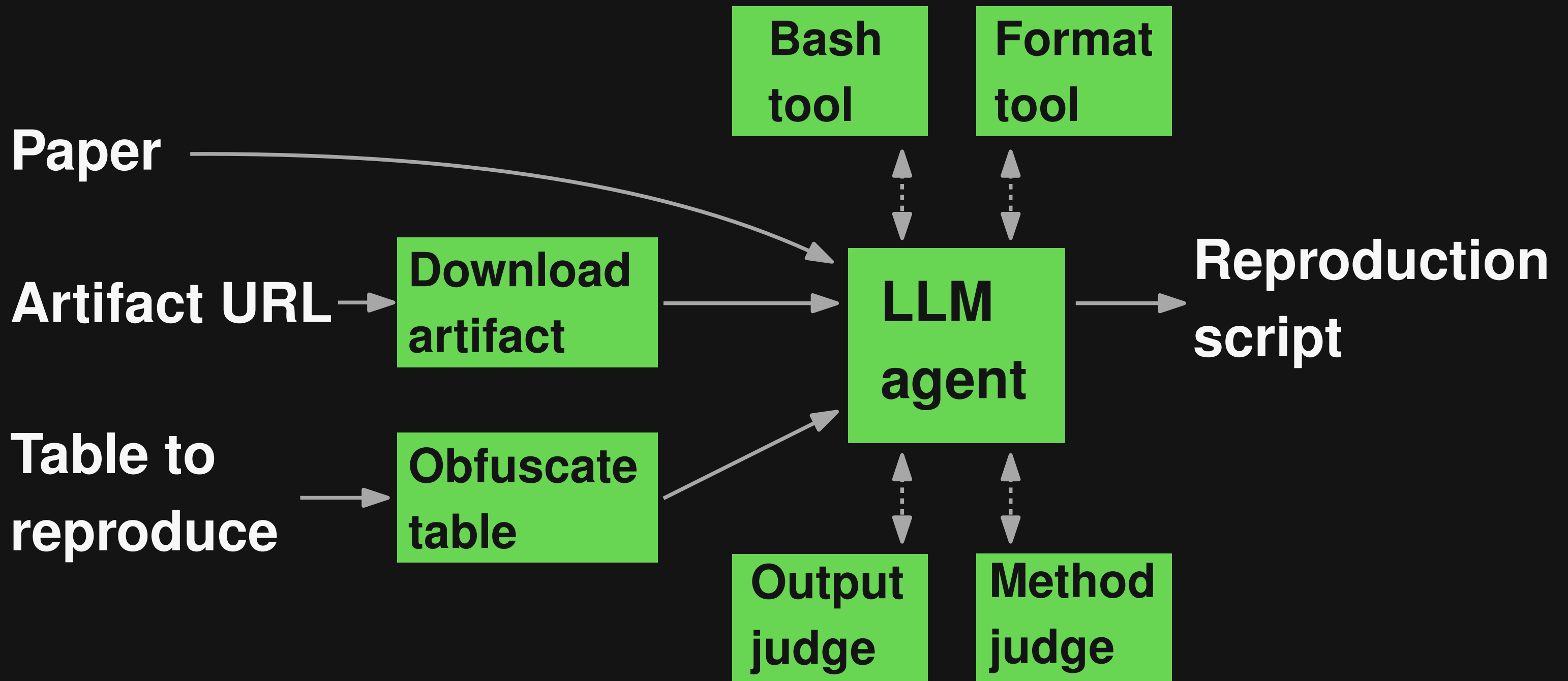
Table to
reproduce



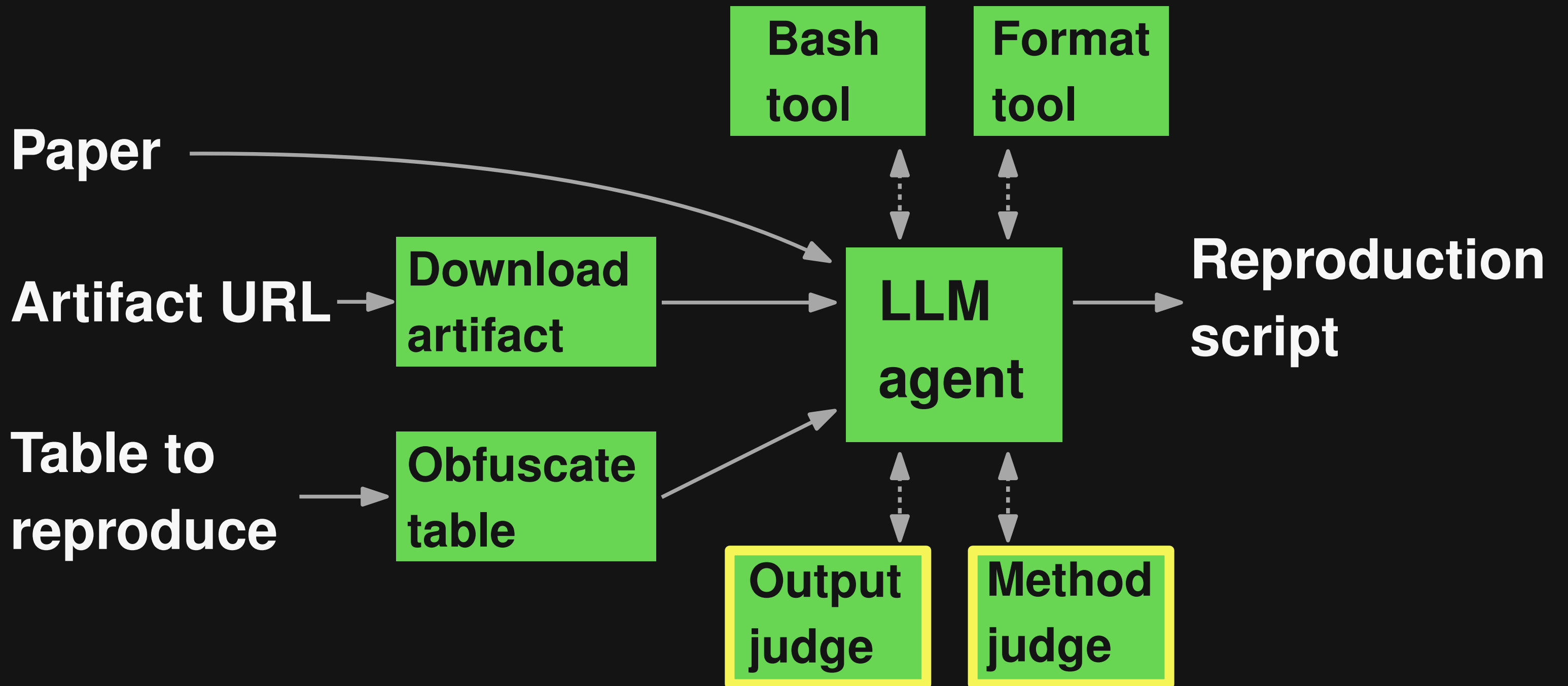
Artisan

**Reproduction
script**

Artisan: Overview



Artisan: Overview



Artisan: Validation Mechanisms

Output judge

- Compare predicted table to ground truth
- Provide partially obfuscated table as feedback

Method judge:

LLM to classify reproduction script into

- Copied results
- Last-mile reproduction
- Full reproduction

Artisan: Validation Mechanisms

Output judge

- Compare predicted table to ground truth
- Provide partially obfuscated table as feedback

Method judge:

LLM to classify reproduction script into

- Copied results
- Last-mile reproduction
- Full reproduction

Simply copies pre-computed, checked-in results

```
cat artifact/summaries/summary_results_fingerprint.txt > /workspace/repro.txt
```

Artisan: Validation Mechanisms

Output judge

- Compare predicted table to ground truth
- Provide partially obfuscated table as feedback

Method judge:

LLM to classify reproduction script into

- Copied results
- Last-mile reproduction
- Full reproduction

Lightweight reproduction from pre-computed, raw results

```
docker-compose run pmsat python parse_all_results_timeouts.py  
↔ benchmarkingset-rc2-results > /workspace/repro.txt
```

Artisan: Validation Mechanisms

Output judge

- Compare predicted table to ground truth
- Provide partially obfuscated table as feedback

Method judge:

LLM to classify reproduction script into

- Copied results
- Last-mile reproduction
- Full reproduction

Full run of experiment

```
docker-compose run pmsat /bin/bash -c "cd /pmsat-inference && python -m pip  
↪ install 'numpy<2' && python run_pmsat_on_traces.py  
↪ examples-results/ping_pong_example/info.json -nmax 7 && MPLBACKEND=Agg  
↪ python parse_single_run_results.py  
↪ TRACE-results/92e710ef352c4739cd7569794272588a" > /workspace/repro.txt
```

Artisan: Validation Mechanisms

Output judge

- Compare predicted table to ground truth
- Provide partially obfuscated table as feedback

Method judge:

LLM to classify reproduction script into

- Copied results
- Last-mile reproduction
- Full reproduction

Reject → continue agentic loop

Accept

Artisan: Results

60 tables from 23 SE papers

	Success		Failure				
	Full reprod.	Last-mile reprod.	Copied-result	Mismatch	Runtime err.	Static err.	
SWE-agent	0	3	4	15	8	30	
OpenHands	5	7	4	12	5	27	
mini-swe-agent	9	5	12	22	9	3	
Artisan	24	20	2	10	4	0	

Artisan: Results

60 tables from 23 SE papers

	Success		Failure				
	Full reprod.	Last-mile reprod.	Copied-result	Mismatch	Runtime err.	Static err.	
SWE-agent	0	3	4	15	8	30	
OpenHands	5	7	4	12	5	27	
mini-swe-agent	9	5	12	22	9	3	
Artisan	24	20	2	10	4	0	

Beyond successful reproduction:

Found 20 errors in published papers & artifacts

Conclusion

- **Specialized agents** successfully complete complex, multi-step workflows
 - Task-specific stages
 - Specialized tools
 - Validation mechanisms
- **Open questions**
 - Better agents vs. better models?
 - How to create specialized agents automatically?