undo

You're Absolutely Wrong!

How to solve complex problems in C++ code - with AI

Everyone knows that debugging is twice as hard as writing a program in the first place.

So if you're as clever as you can be when you write it, how will you ever debug it?

Brian Kernighan

Everyone knows that debugging is twice as hard as writing a program in the first place.

So if your AI is as clever as it can be when it writes it, how will it ever debug it?

~~*Brian Kernighan*~~

undo

LLMs and debugging

Lots of talk about LLMs finding bugs you didn’t know about (e.g. Mythos)

This talk is about finding the root-cause of something that is hurting you right now

- The agent is iterating having just produced some buggy code
- You’ve integrated beautiful AI debug triage into your CI
- Maybe this bug has been in the code base for ages

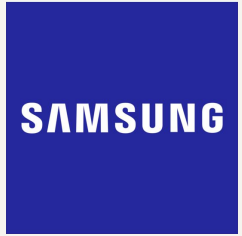
Always has been
Wait, all that time is still spent debugging?
imgflip.com

UNDO HAS SPENT 20 YEARS SOLVING THIS PROBLEM

undo

EDA & Computational Software

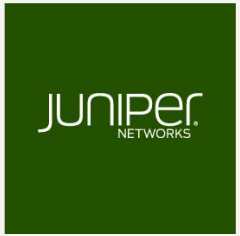
Semiconductors & Silicon Design



Financial Services

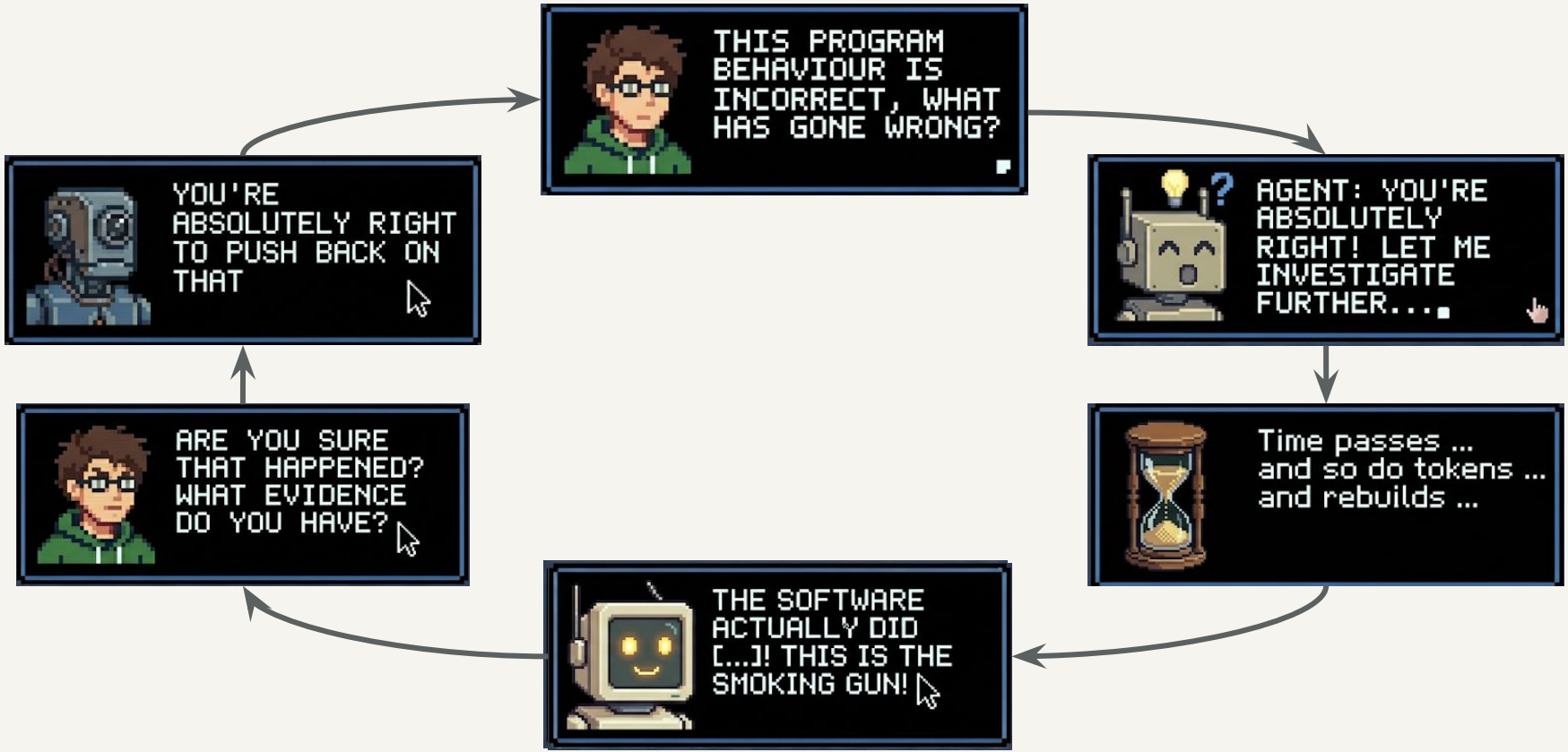
Data Management

Networking & Security



undo

Debugging is hard: 2026 Edition



undo

What is AI good at?

- Ingesting large amounts of (semi)-unstructured data
- Identifying patterns
- *Generating hypotheses*
- Exploring code

undo

What is AI bad at?

- Knowing what *you* know
  - E.g. I’ve seen something before => look over here
- Knowing what truth looks like
  - i.e. common sense
- Telling you what information it *lacks*

undo

Give it context & ask for evidence

- **Context** is a limiting factor on how clever your agent can be
- In both senses - the *context window* and *information about the problem*
- For best results:
  - Give it all the context you have
  - Give it a way to get more context
  - **But don’t give it lots of irrelevant context**

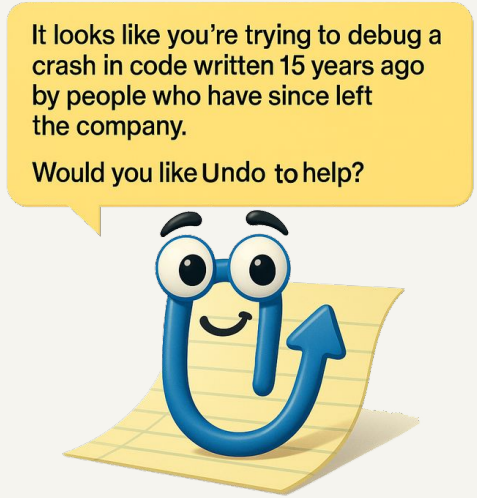
We don’t just want debugger clippy

Not a chat interface onto a debugger

“Please put a breakpoint on line 42”

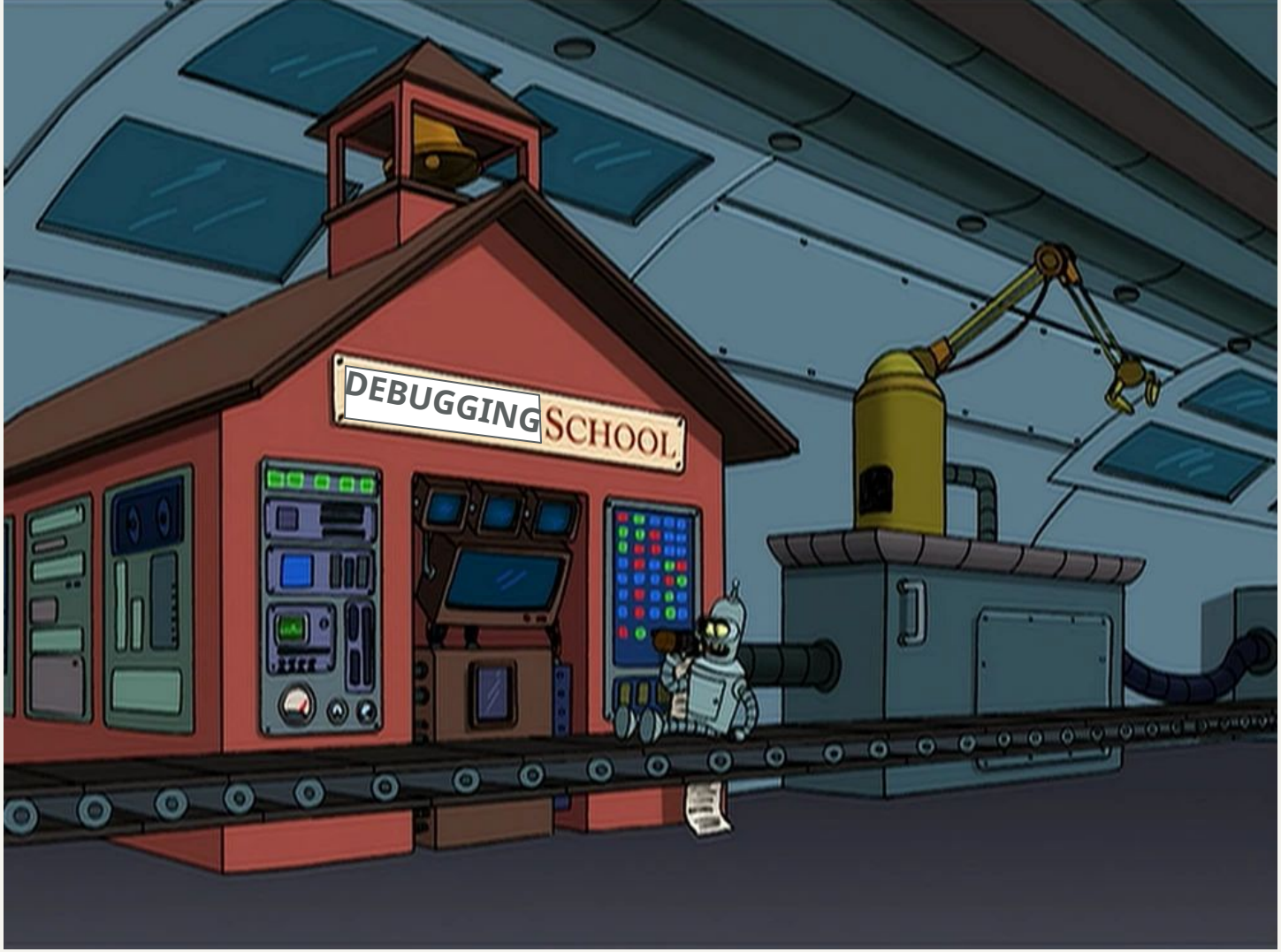
But an end-to-end fully automated root cause analysis

“Why did this program produce the result 42?”



AI = Model + Context

BASELINE: EDUCATE YOUR AGENT


DEBUGGING SCHOOL

undo

AGENTS.md

- Agents read this when working in a directory containing it
- Top-level README content for that directory
  - How to run tests, coding style, project structure, etc
  - Can refer to other files or skills for more in-depth info
- Ask your agent to make one, then customise it!
- Version control this to share with your team
  - Or use `AGENTS.local.md` so nobody sees your crazy setup

undo

Skills

- Guess what - another (specially structured) Markdown file!
- Efficient on context - details loaded on demand
- Gives additional information about how to do something:
  - Use a particular tool
  - Perform a particular task
- May include *scripts* to be run whilst the skill is in use

undo

How we teach our coding agent @ Undo

Top-level

`AGENTS.md` / `CLAUDE.md`

"Required reading"

- Architecture guide
- How to build & test
- Top-level style guide

Skills

`.claude/skills/<name>/`

How to accomplish a task

- Write code in module X
- Triage a test failure
- Update the bug tracker

Points to relevant docs and provides a checklist of actions

Docs

`docs/internal/`

Design reference

- Source of truth
- For agents *and* humans

The best store for general purpose knowledge

undo

Agentic Reasoning & Tool Use

Prompts
What the user asked for *this time*

AGENTS.md / CLAUDE.md
Top-level instructions for this project

Skills
How to accomplish tasks

Other sources of info
Docs, specs, bug tracker...

Guides

LLM

Reasoning

Uses

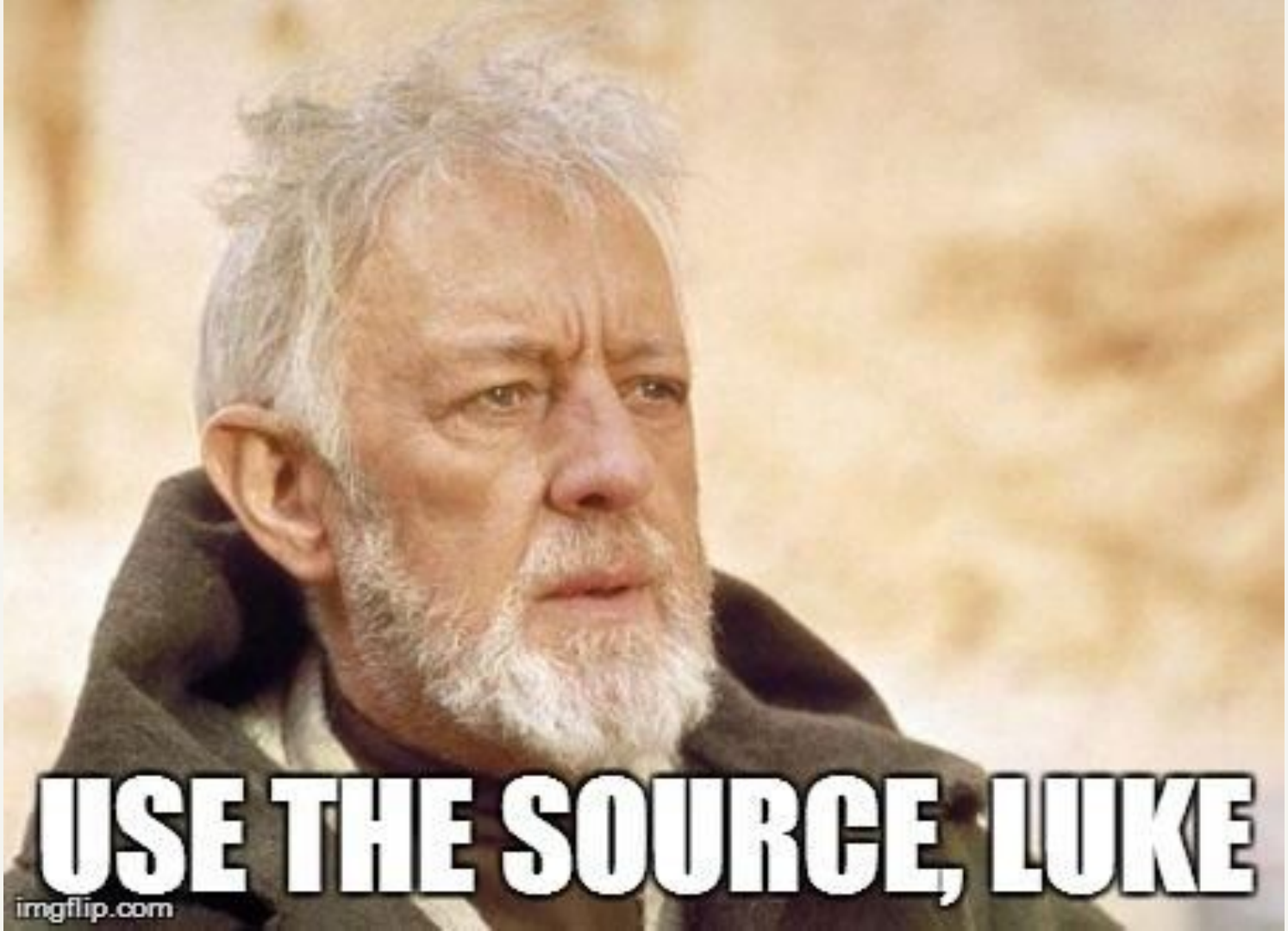
Built-in tools
Provided directly by the agent front-end

CLI tools
Invoked directly - typically in a one-shot fashion, maybe within a script

Custom tools
Generated by the agent for a specific purpose in e.g. Python

MCP servers
Local or remote - useful for longer running sessions with stateful behaviour

APPROACH 1: THE ANSWER'S IN THE CODE


USE THE SOURCE, LUKE
imgflip.com

undo

Static analysis

- Agents are increasingly good at this kind of reasoning
- But when you're looking for a bug:
  - There's a huge possibility space to consider
  - Can burn a lot of tokens whilst exploring + reasoning
- Better to have rich *runtime context* - information about how the program *behaved* when running
- Without this, there's no ground truth to guide the above *nor* to sanity check the answer

undo

Aside: Fable 5 - Does it need runtime context?

- Sample issue from our internal benchmarks: DuckDB (~0.5M LOC) - Issue #20086
- Without Runtime Context
  - Hallucinated result, taking over 1 hour
  - Set $37.60 of tokens on fire 🔥
- With Runtime Context (various forms tried)
  - Root cause within 12 - 24 minutes
  - Cost from $2.88 to $6.31

undo

Runtime context for beginners: Give it a ~~backtrace~~!

- Copy-pasting a backtrace (from crash or assertion failure) is basic runtime context
  - At the time of the failure, what was going on?
  - *Implies* a little bit of *history* - how did we get here?
- Kick-starts source code analysis and gives some grounding

HEY CLAUDE
38
WHAT CAUSED THIS BACKTRACE?
imgflip.com

APPROACH 2: GET THE CODE TO SHOW THE WAY

That way
This way

undo

Let it rebuild and add logging (if you can)

- Agents love writing code
- They're therefore pretty eager to add logging and pretty good at using it sensibly
  - Make sure they know *how* they should (`AGENTS.md` / skills)
- This can be a winning strategy if your builds are *quick* and your bug is deterministic (enough) so going round this loop is cheap

undo

ASAN, TSAN, etc

- Linking a sanitizer is like a cheat code for the bugs it can detect
- Libraries like ASAN and TSAN turn silent failures into errors
- Agents are *also* good at using sanitizers
  - They reach for them quite enthusiastically
  - These also imply a rebuild!
- Again, make sure your agent knows *how* to make an ASAN build in your system (AGENTS.md / skills)

undo

Debug features in agents

- See also Cursor's debug mode: https://cursor.com/blog/debug-mode
- Scaffolding to drive the LLM through a debugging process: gathering hypotheses, getting evidence human-in-the-loop verification, etc
- Other agents may add similar behaviour - or could be added via skills...

APPROACH 3: GIVE THE AGENT A DEBUGGER

undo

Core Dumps

- Even better than a backtrace
  - Contains backtraces for all threads
  - Contains all memory (therefore globals, locals and heap)
  - Might be able to spot stack corruption
- Needs a debugger and a way to control it...

GDB (Scripting)

- The simplest debugger integration:
  - Have `gdb` available on your system
  - The agent will write little scripts and run them against a core dump (or real process!)
- These work as one-shot invocations (fire up GDB, attach or start your program, do a thing, shut down)
  - Better suited when iterating is cheap / quick / deterministic

AI in your debugger

- Influential (particularly ChatGDB) early work to extend a conventional debugger with AI functionality
  - https://github.com/jjravi/AgentGDB
  - https://github.com/pgosar/ChatGDB
- Better thought of as “An assistant to help use the debugger”
- Able to ask (simple) questions about program state or request debugger actions in natural language

undo

Debugger in your agent (MCP Servers)

- GDB https://github.com/Ipiano/gdb-mcp
- WinDBG https://github.com/svnscha/mcp-windbg
- VS Code https://github.com/microsoft/DebugMCP
- Binary Ninja https://github.com/fosdickio/binary_ninja_mcp
- LLDB https://github.com/stass/lldb-mcp

undo

Aside: MCP design is not simple

- It’s tempting to imagine that just exposing as much functionality as possible to the agent is good
- BUT
  - Agents have a fixed reasoning budget to spend - better to spend it on the problem than the tool
  - Irrelevant functionality gives the agent ways to confuse itself
- More flexibility may reduce autonomy - consider guidance via skills if you’re using a very “bare” integration

APPROACH 4: GIVE THE AGENT RECORDING CAPABILITY

Agent with recording capability
(artist's impression)

undo

Process Recording!

- Captures a deterministic, instruction-precise execution trace
- Best known:
  - GDB’s built-in process recording
  - WinDBG TTD
  - RR
  - Undo

undo

MCP is really useful when you are building this!

- A recording eliminates the need to rebuild / re-run - just query whatever is needed
- BUT replaying benefits from caching its state so it’s better not to keep restarting the tool
  - For rr: https://github.com/karellen/karellen-rr-mcp
  - For Undo: integrated MCP server

undo

Have the AI compare multiple recordings

This is a super power! Turn “it works for me” to your advantage

- Working vs not working version of the code
- Working vs not working flaky tests
- Fast vs slow

(if you don’t have recording available, use the comparison trick on whatever runtime context you *do* have)

SUMMARY

undo

Debugging in an AI world

Debugging has always been and continues to be a major part of software development

AI's can be good at debugging but only if you drive them right

- `SKILL.md`, `AGENTS.md`, etc

Context is vital: the static (code, docs) and the runtime (what happened)

MCP is a great way to get the runtime context into the agent

Recordings are the ultimate source of runtime context

VISIT UNDO.IO TO LEARN MORE