

# Substrate Engineering

Engineering Foundations *in a World of* LLMs

Chris Krycho – StaffPlus New York 2024



language:C language:C++

## Filter by

<> Code 0

**Repositories 3.4M**

Issues 10M

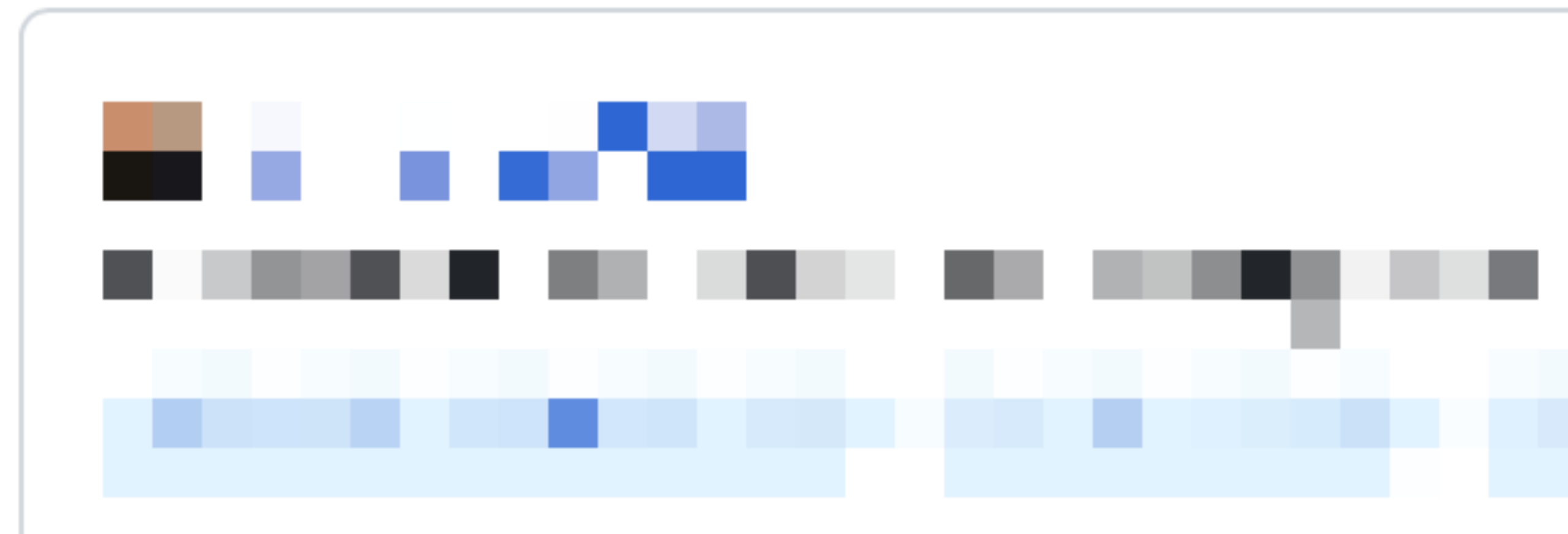
Pull requests 13M

Discussions 0

Users 2M

More

3.4M results (1 s)



# Substrate Engineering

*Who do you trust to write memory- and thread-safe code in C?*

- Yourself?
- A new junior on your team?
- GitHub Copilot?

# Substrate Engineering

*Who do you trust to write memory- and thread-safe code in (safe) **Rust**?*

- Yourself?
- A new junior on your team?
- GitHub Copilot?

Our systems are *not ready* for  
a world of pervasive LLMs.

If you are a *big fan* of LLMs:  
this talk is about **MAKING THEM BETTER.**

If you are *skeptical* of LLMs:  
this talk is about **GOOD SAFEGUARDS.**

# Prompt engineering

*An emerging discipline?*

- Specific choices in wording.
- The amount of context to include.
- The scope of the task you are giving it.
- Creativity levels.
- What not to bother with because it tends to go sideways there.
- Meta prompts, like my favorite: including “no blabbing”.



# Problem

LLMs have been trained on *real-world code*.

Prompt engineering *is not enough.*

Prompt engineering *will* **NEVER** *be enough*.

*Why?*

A *substrate* is a layer that sits *below* the thing we are interested in.

# Substrates

## *Biology*

- Where an organism grows
- Possibly what it eats
- Where it lives

Everything about its existence!

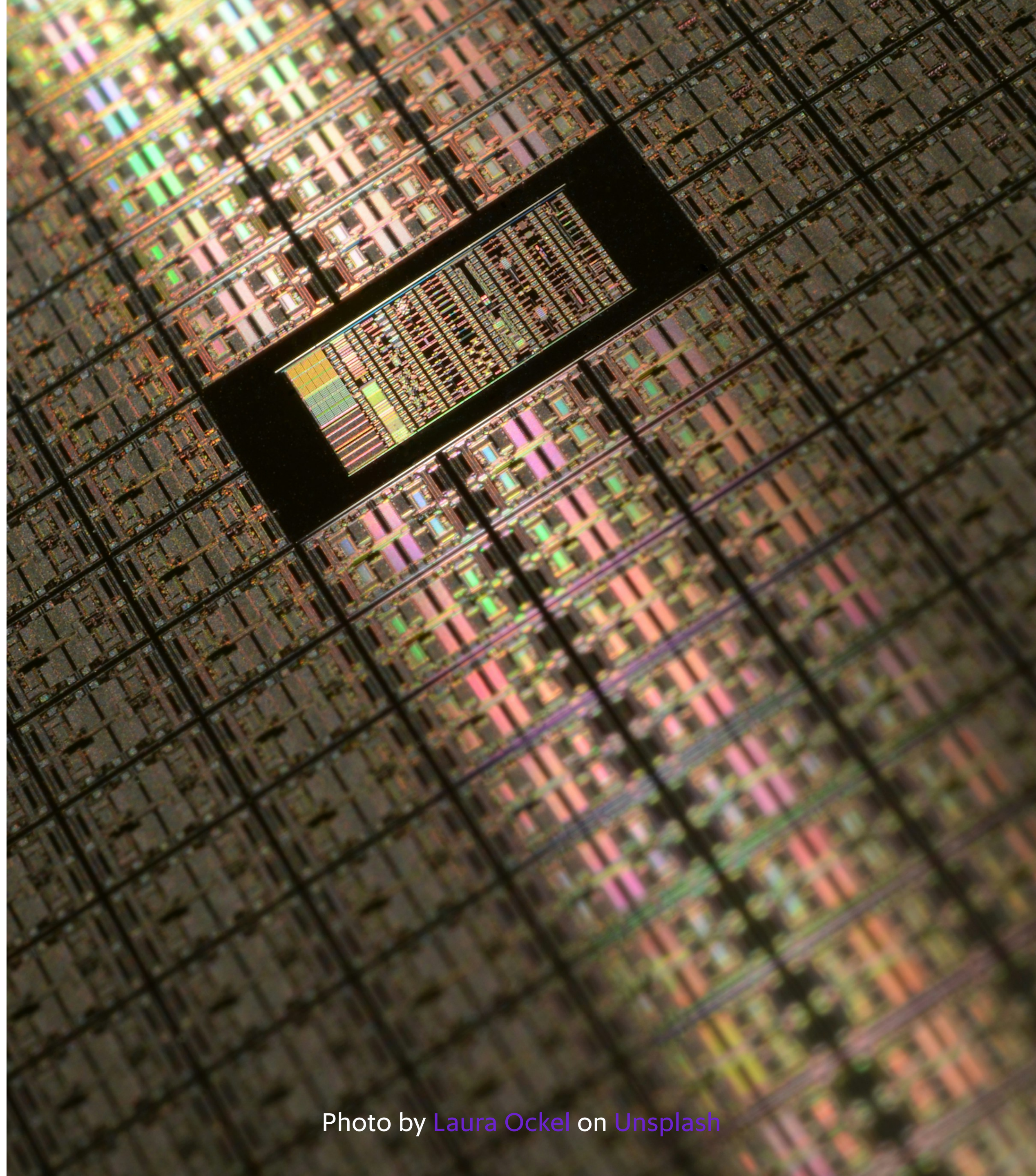


# Substrates

## *Chip manufacturing*

Silicon wafer: "does nothing".

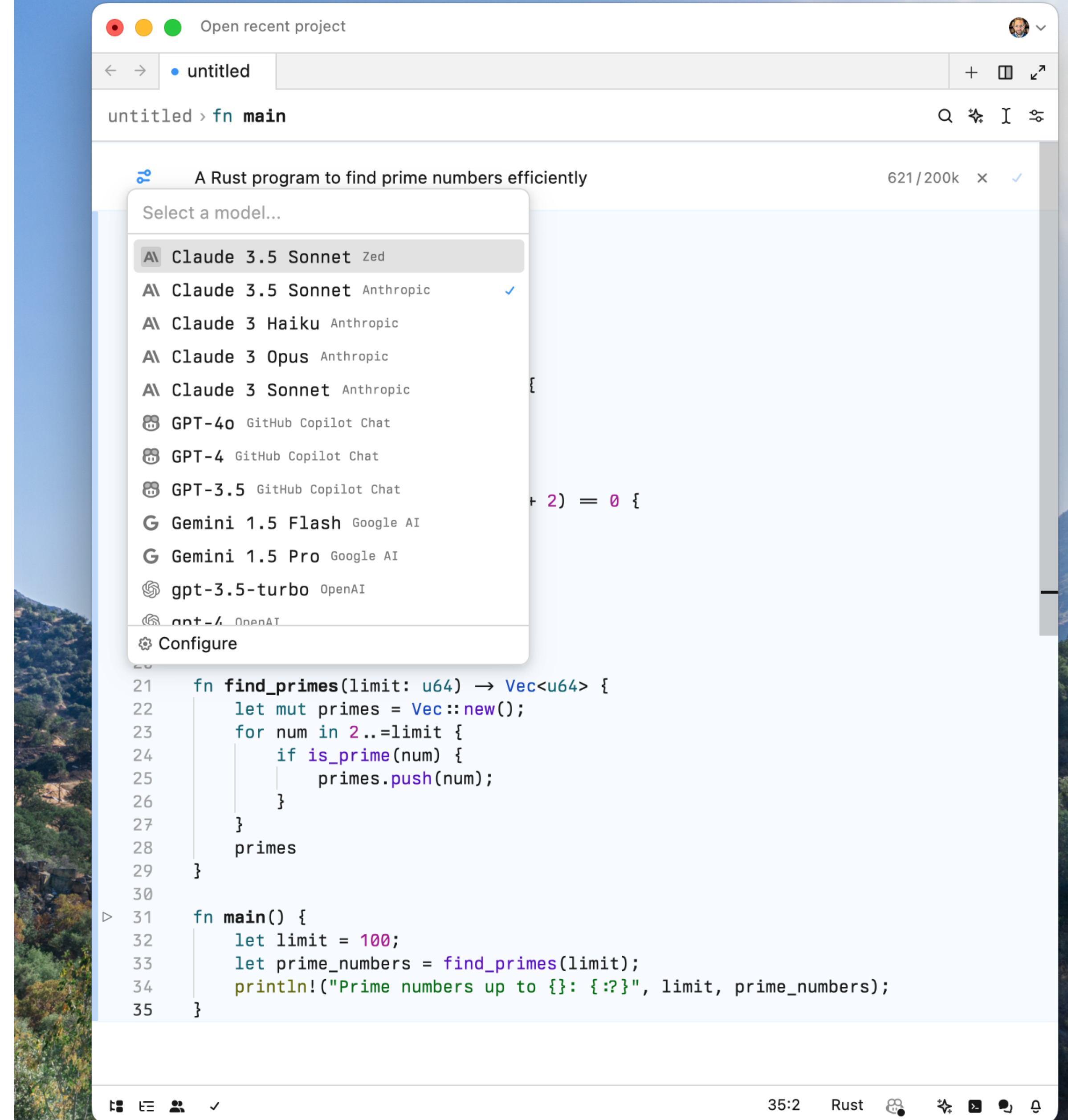
But no wafer? No useful chip.



# Substrates

## *Large language models*

- Training data (input)
- Engineering systems (output)





# Automation *and* Attention

# Automation *and* Attention



Photo by [Fabio Romano](#) on [Unsplash](#)



Photo by [Roberto Nickson](#) on [Unsplash](#)

The *better* **AUTOMATION** works,  
the *less* we **ATTEND** to it.

# Automation and Attention



Photo by [Fabio Romano](#) on [Unsplash](#)



Photo by [Roberto Nickson](#) on [Unsplash](#)

A screenshot of the GitHub Copilot extension page on GitHub. The page features the GitHub Copilot logo, the name "GitHub Copilot" with version "v1.223.0", and a star rating of 1066. Below this, there are links for "Install" and "Auto Update". The main heading is "GitHub Copilot - Your AI pair programmer". A description states: "GitHub Copilot is an AI pair programmer tool that helps you write code faster and smarter." There is a link to "Sign up for a GitHub Copilot free trial". The bottom part of the screenshot shows a code editor with Python code for a Flask application, including routes for creating, updating, and deleting todos.

**GitHub Copilot** v1.223.0  
GitHub [github.com](#) | 19,330,699 | ★★★★★ (1066)  
Your AI pair programmer  
[Install](#)  Auto Update

DETAILS FEATURES CHANGELOG EXTENSION PACK

## GitHub Copilot - Your AI pair programmer

GitHub Copilot is an AI pair programmer tool that helps you write code faster and smarter.

[Sign up for a GitHub Copilot free trial](#)

```
def create_todo():
    id = str(uuid.uuid4())
    title = request.form['title']
    completed = False
    todos.append(newTodo)
    return redirect('/')

@app.route('/update-todo/<id>')
def update_todo(id):
    todo = [todo for todo in todos if todo['id'] == id][0]
    todo['completed'] = not todo['completed']
    return redirect('/')

@app.route('/delete-todo/<id>')
def delete_todo(id):
    todo = [todo for todo in todos if todo['id'] == id][0]
    todos.remove(todo)
    return redirect('/')

if __name__ == '__main__':
    app.run()
```

# Challenge

CODE REVIEW *and* DEBUGGING

The *better* LLMs get—  
the *more* they **BOOST VELOCITY**,  
by generating **WORKING CODE**—  
the *harder* it will be to notice when  
they get things **WRONG**.

# Automation *and* attention

*What do we do?*

— “Defense in depth” for software foundations

# Automation *and* attention

*What do we do?*

- “Defense in depth” for software foundations
- Judgment about where LLMs should and should not be allowed



# Problem

**HALLUCINATION**

**HALLUCINATION** is not a *solvable problem*.

**HALLUCINATION** is the *wrong word*.

**HALLUCINATION** is *just what* LLMS *are*.

We can (and *must*) build our  
*software* and *social* **SYSTEMS** accordingly.

# Substrate Engineering

# Substrate Engineering

## *Key constraints*

- Substrate/environment
- Automation and attention
- How LLMs actually work

# Substrate Engineering

## *The territory*

- Tooling and Configuration
- Languages
- API design
- Testing
- Agents
- Package managers
- Operating systems



# Substrate Engineering

*The territory we have time to cover*

— Tooling and Configuration

# Tooling *and* Configuration

# Tooling *and* Configuration

a **DISPROPORTIONATE IMPACT** on  
user & developer experience

# Tooling *and* Configuration

*Now add LLMs into the mix.*

These kinds of tools and configuration languages are:

- Extremely **amenable** to use with LLM-based systems
- Extremely **vulnerable** to the failure modes of LLMs

# Tooling *and* Configuration

*Now add LLMs into the mix.*

These kinds of tools and configuration languages are:

- Extremely ***amenable*** to use with LLM-based systems
- Extremely *vulnerable* to the failure modes of LLMs

# Tooling *and* Configuration

*Now add LLMs into the mix.*

These kinds of tools and configuration languages are:

- Extremely *amenable* to use with LLM-based systems
- Extremely **vulnerable** to the failure modes of LLMs

# Tooling *and* Configuration

*A broken GitHub Actions config*

```
jobs:  
  - name: Test  
    runs-on: ubuntu-latest  
    steps:  
      - name: Check out repository code  
        uses: actions/checkout@v4  
      - run: make test  
on: [push, release]
```

# Tooling *and* Configuration

*A broken GitHub Actions config*

```
jobs:  
  - name: Test  
    runs-on: ubuntu-latest  
    steps:  
      - name: Check out repository code  
        uses: actions/checkout@v4  
      - run: make test  
on: [push, release]
```

> "jobs" section is sequence node but mapping node is expected



# Tooling *and* Configuration

*A fixed GitHub Actions config*

```
jobs:  
  Test:  
    runs-on: ubuntu-latest  
    steps:  
      - name: Check out repository code  
        uses: actions/checkout@v4  
      - run: make test  
on: [push, release]
```

# Tooling *and* Configuration

*A broken GitHub Actions config*

```
jobs:  
  - name: Test  
    runs-on: ubuntu-latest  
    steps:  
      - name: Check out repository code  
        uses: actions/checkout@v4  
      - run: make test  
on: [push, release]
```

> "jobs" section is sequence node but mapping node is expected

# Tooling *and* Configuration

*Investing in ops* **LANGUAGES**

— Pulumi: TS, Python, .NET, Java, Go

— F#'s FAKE DSL

...unlocks the power and tooling of “full” programming languages.

# Problem

“Full” programming languages can do *anything*.

# Tooling *and* Configuration

*Investing in ops* **LANGUAGES**

- Infinite loops during installation
- `undefined` is not a function during deployment
- Throwing `java.lang.NullPointerException` in CI

# Tooling *and* Configuration

*Investing in ops* **LANGUAGES**: *useful properties*

- Soundness:
  - no undefined is not a function
  - no `NullPointerException`s

# Tooling *and* Configuration

*Investing in ops* **LANGUAGES**: *useful properties*

- Soundness
- Termination: guaranteeing the program will end

# Tooling *and* Configuration

*Investing in ops* **LANGUAGES**: *useful properties*

- Soundness
- Termination: guaranteeing the program will end
  - totality: every input has an output, even things like  $n/0$ 
    - purity: same input = same output, no side effects
  - no general recursion: no `while (true) { ... }` or equivalents.



# Tooling *and* Configuration

*Investing in ops* **LANGUAGES**: *useful properties*

- Soundness
- Termination
- Rich type system
  - Discriminated union/sum/algebraic data types  
`type Track = IndividualContributor | Manager`
  - Guaranteed inference

# Tooling *and* Configuration

*Investing in ops* **LANGUAGES**: *useful properties*

- Soundness
- Termination
- Rich type system (but not *too* rich)
  - Discriminated union/sum/algebraic data types  
`type Track = IndividualContributor | Manager`
- Guaranteed inference

# Tooling *and* Configuration

*Investing in ops **LANGUAGES**: the wins*

- Faster feedback loops for people writing configuration
  - With or without LLMs!
- LLM training data would be much more correct




A much higher chance of getting it right from the outset.

Not a silver bullet.

But having the right tools in the toolbox matters.







# Tooling *and* Configuration

*Investing in ops* **LANGUAGES**: *candidate languages*

- Starlark (née Skylark): build language used by Bazel and Buck/Buck2
  - Soundness 
  - Termination 
  - Rich type system (but not *too* rich) 






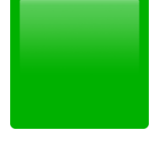
# Tooling *and* Configuration

*Investing in ops* **LANGUAGES**: *candidate languages*

- Starlark (née Skylark): build language used by Bazel and Buck/Buck2
  - Soundness 
  - Termination 
  - Rich type system (but not *too* rich) 
- Dhall
  - Soundness 
  - Termination 
  - Rich type system (but not *too* rich) 

# Tooling *and* Configuration

*Investing in ops* **LANGUAGES**: *candidate languages*

- Starlark (née Skylark): build language used by Bazel and Buck/Buck2
  - Soundness 
  - Termination  *Increasingly widely used*
  - Rich type system (but not *too* rich) 
- Dhall
  - Soundness 
  - Termination  *No one has heard of it*
  - Rich type system (but not *too* rich) 

# Tooling *and* Configuration

*Investing in ops **LANGUAGES**: more work needed!*

What if we built a language like Starlark *plus* Dhall—familiar but robust?

- Does it work well in practice?
- What are the tradeoffs?
- Investigation needed!

# Conclusion



Put *all* of our software engineering on  
**BETTER FOUNDATIONS**

**ENGINEERING FOUNDATIONS** are the  
**SUBSTRATES** for *all* software engineering

# Thank you!

*I appreciate your attention.*

- Read: [chriskrycho.com](https://chriskrycho.com)
- Email: [hello@chriskrycho.com](mailto:hello@chriskrycho.com)
- Follow: [\(@\)chriskrycho\(.com\)](#)
- Calendly:

