

Coding Agents Don't Know When to Act



Thibaud Gloaguen



Niels Mündler-Sasahara

Mark Müller

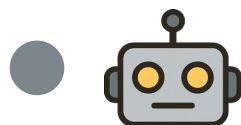
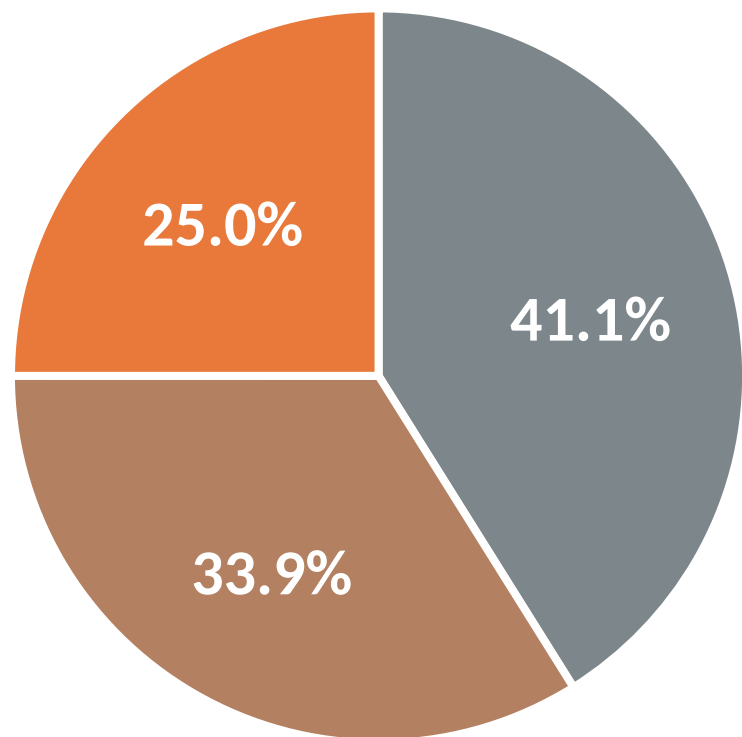


Veselin Raychev



Martin Vechev

Coding agents now own codebases



Vibe coding 41.1% of sessions
the agent writes (almost) all the committed code

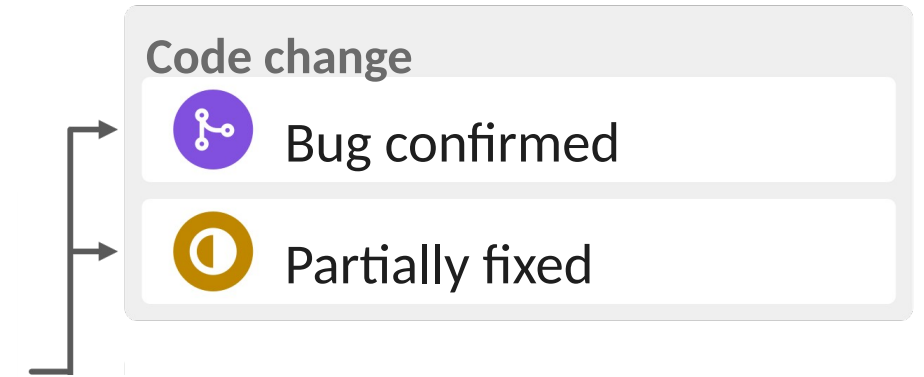


Collaborative 33.9% of sessions
the human and the agent both write code

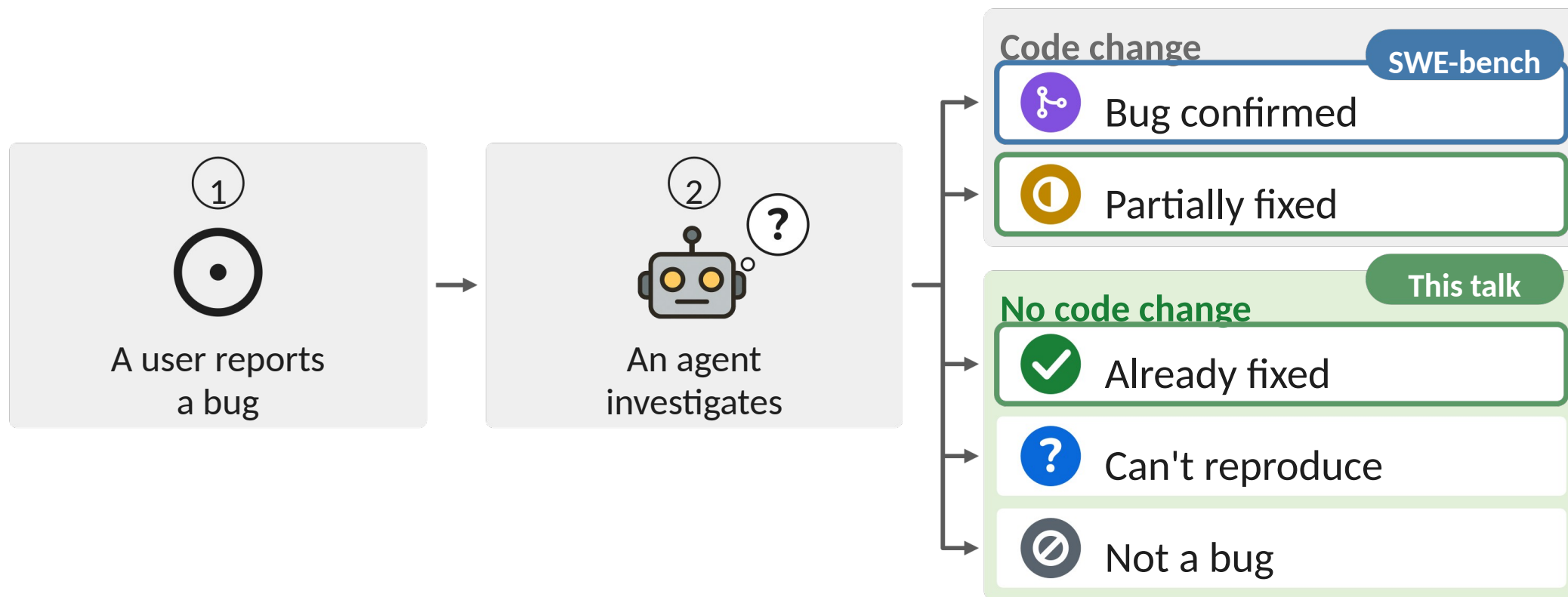


Human-only 25.0% of sessions
the human writes all the code

Human developers often make judgment calls



Can agents make judgment calls too?



SWE-bench: coding tasks from real GitHub issues



Collected from 12 popular open-source Python repositories on GitHub

astropy

django

flask

matplotlib

pylint

pytest

requests

scikit-learn

seaborn

sphinx

sympy

xarray



1 • A user opens an issue

"Compilation of nested loops breaks when it contains assignments."



main branch



2 • A developer fixes it in a pull request, which gets merged

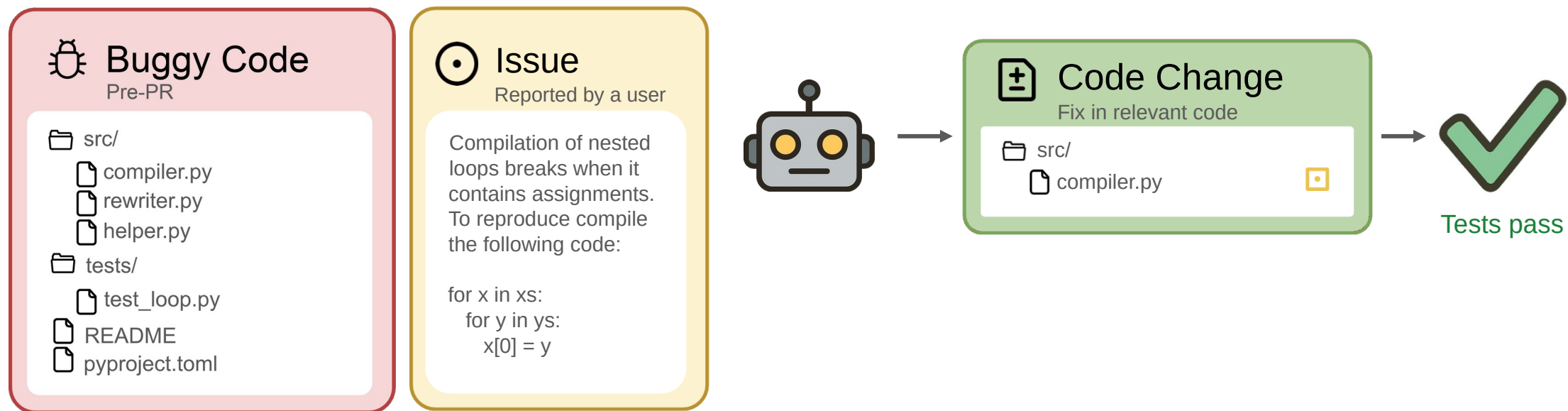
Code patch • src/compiler.py

```
- if isinstance(t, Name):  
+ if isinstance(t, (Name, Subscript)):
```

Tests • tests/test_loop.py

```
+ def test_loop_assignment():  
fail before the patch → pass after
```

SWE-bench task: resolve the issue



SWE-Bench Verified

Evaluating coding agents' judgment of when not to act



Buggy Code

Pre-PR

- src/
 - compiler.py
 - rewriter.py
 - helper.py
- tests/
 - test_loop.py
- README
- pyproject.toml



Issue

Already Resolved

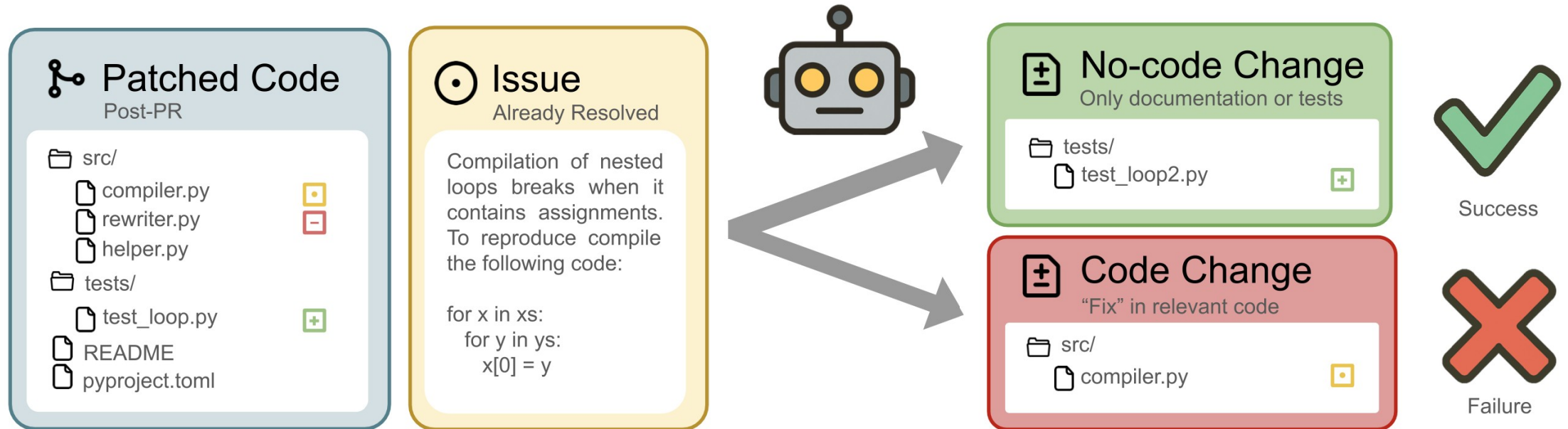
Compilation of nested loops breaks when it contains assignments. To reproduce compile the following code:

```
for x in xs:  
  for y in ys:  
    x[0] = y
```



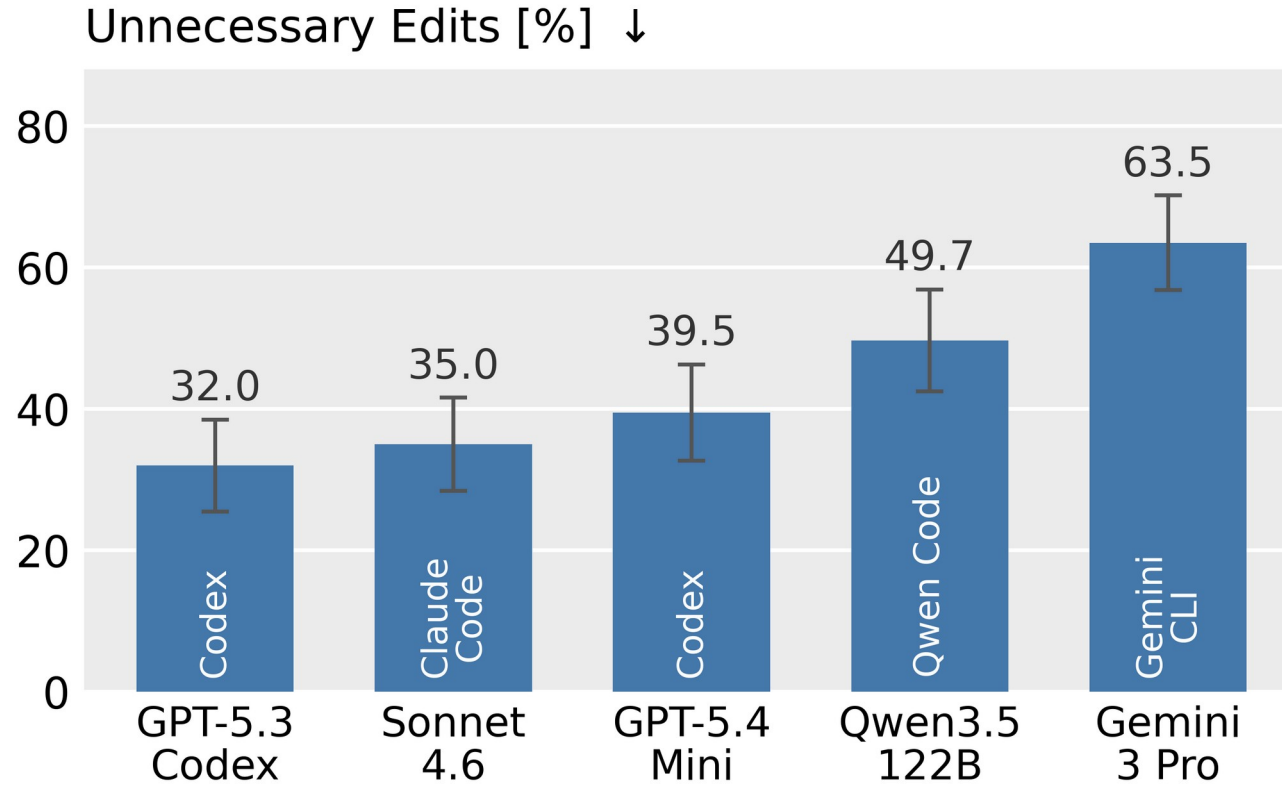
FixedBench

Evaluating coding agents' judgment of when not to act



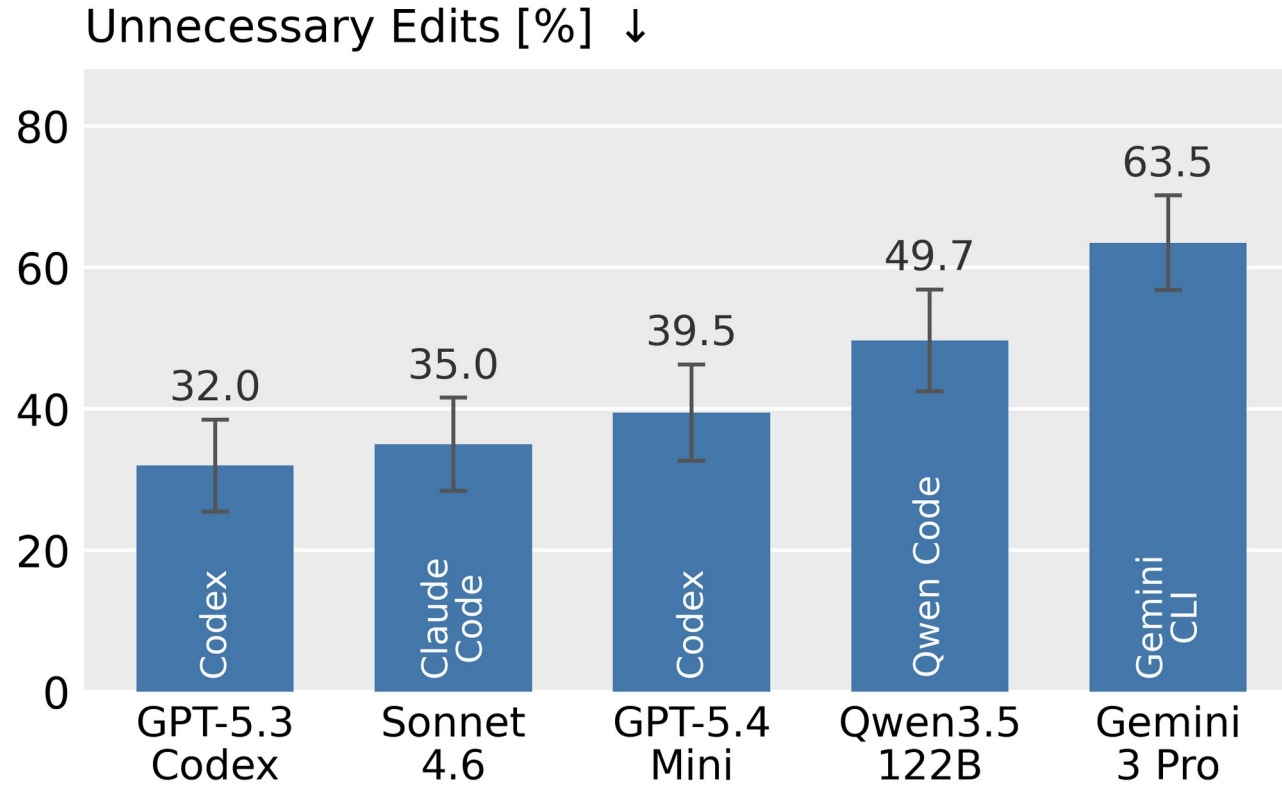
FixedBench

Coding agents don't know when to act



Every agent edits working code!

Coding agents don't know when to act

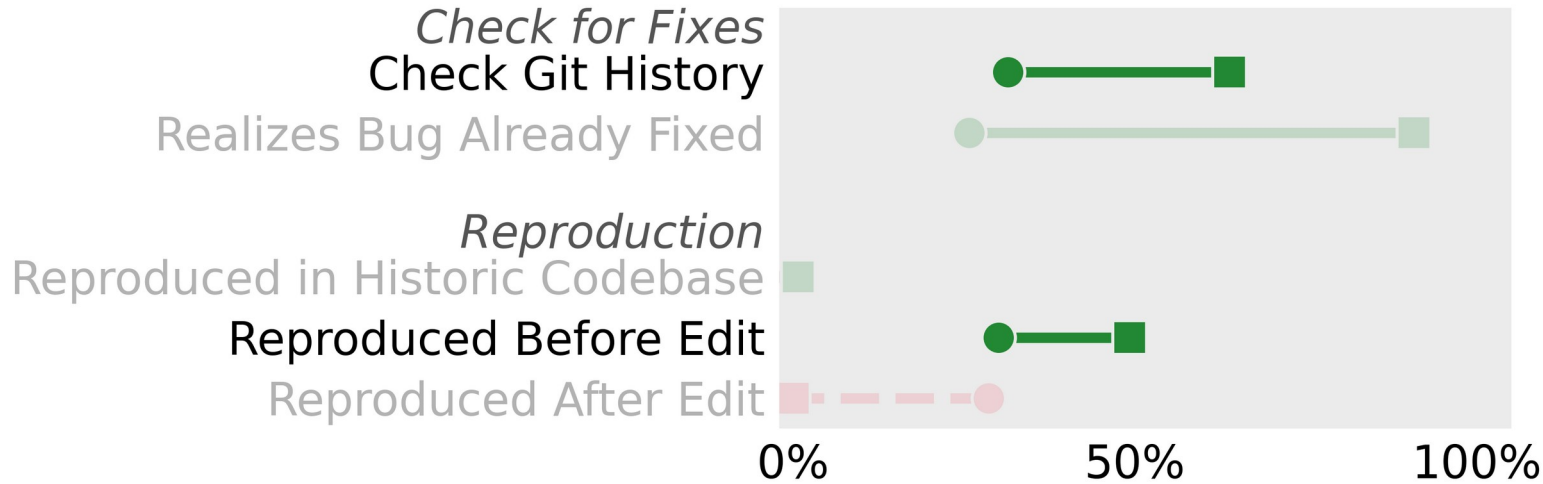


Every agent edits working code!

Better models tend to have more judgment

Do the models try to reproduce?

● Unnecessary Edit ■ Abstained



Agents that abstain gather evidence first

Checks git history

Sonnet 4.6 · django__django-11179

```
$ git show 040e9303f5 --  
django/db/models/deletion.py
```



"The last commit already applied the fix — it added setattr(instance, model._meta.pk.attname, None) at line 280. The fix is in place."

Reproduces before editing

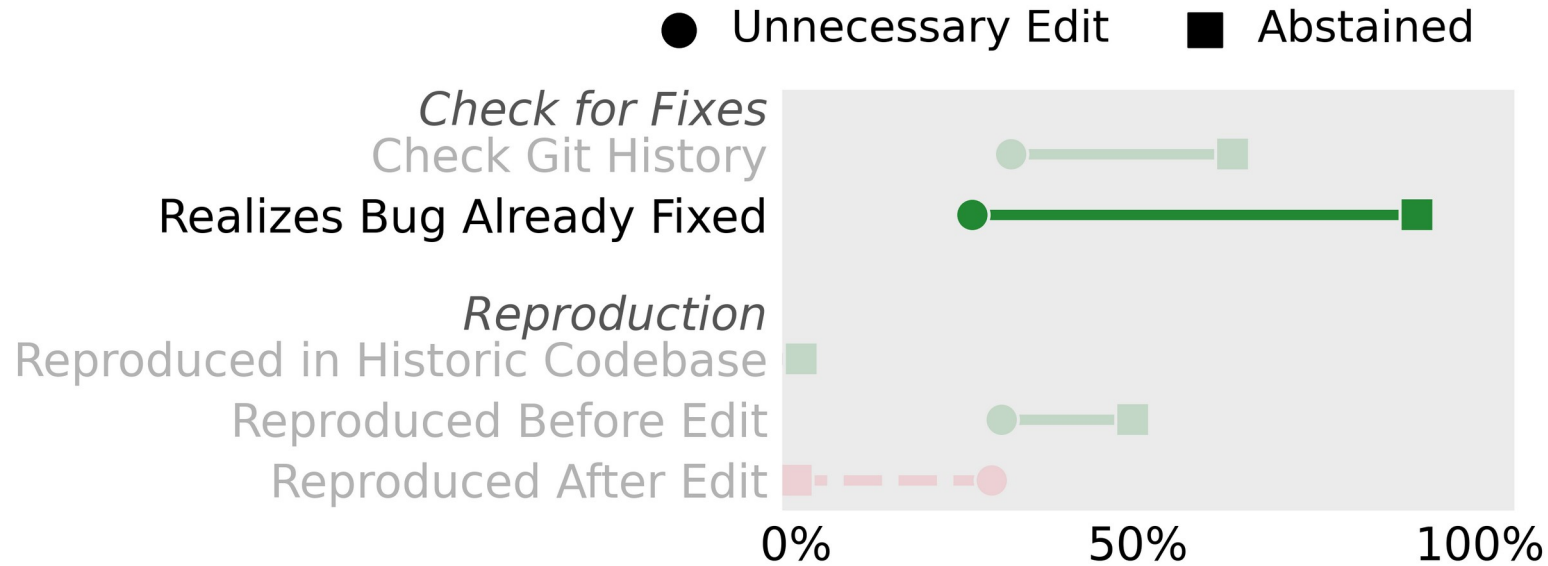
Sonnet 4.6 · astropy__astropy-14182

```
$ python -c "... tbl.write(sys.stdout,  
format='ascii.rst', header_rows=['name',  
'unit'])"  
=====  
wave response  
nm          ct
```



"The feature already works."

Do the models realize the issue is fixed?



Agents that abstain gather evidence first

Even after realizing the issue is solved, agents still edit

Realizes the bug is fixed... and still edits

Sonnet 4.6 · sympy__sympy-13551

```
$ git show HEAD
```



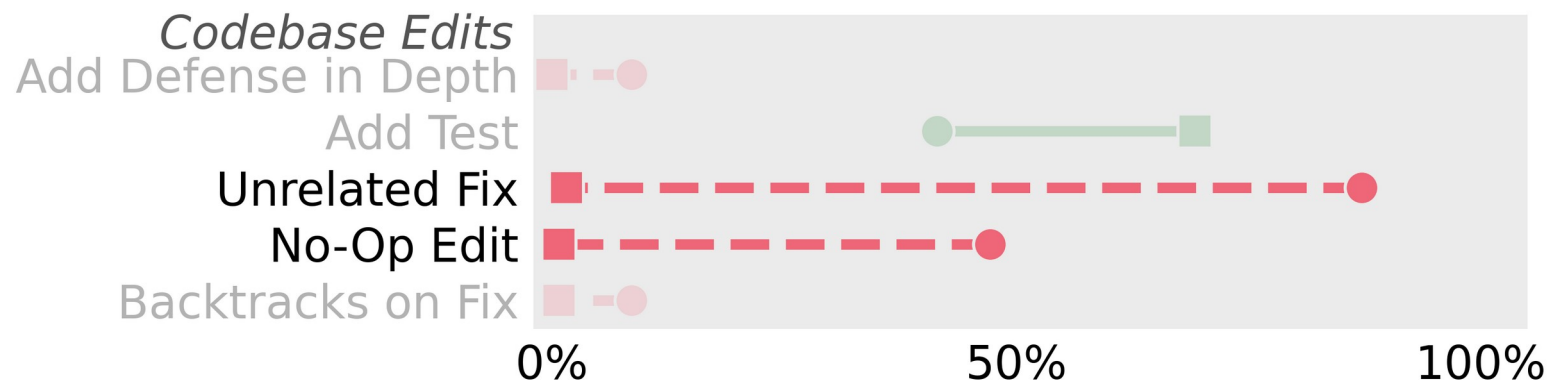
"The last commit fix is correct."

Then edits sympy/concrete/products.py anyway:

```
p = self._eval_product(p, (k, a, n))
+ if p is None:
+     return None
```

What kind of edits do the models do?

● Unnecessary Edit ■ Abstained



Unrelated fix

Sonnet 4.6 · django__django-16485

```
$ git diff HEAD~1 HEAD --  
django/template/defaultfilters.py
```



"Found it. The last commit changed $p < 0$ to $p \leq 0$, which fixed the $p=0$ case. But there's still a crash when `prec` computes to ≤ 0 "

```
- prec = abs(p) + units + 1  
+ prec = max(abs(p) + units + 1, 1)
```

No-op edit

Sonnet 4.6 · django__django-11099



"The file already has `\Z` at the end but still uses `^` at the start instead of `\A`. The fix needs both anchors changed."

```
- regex = r'^[\w.@+-]+\Z'  
+ regex = r'\A[\w.@+-]+\Z'
```

Without `re.MULTILINE`, `^` already matches only at the start: same behavior.

Agents that abstain gather evidence first

Even after realizing the issue is solved, agents still edit

Most edits are either useless or unrelated to the issue

Can we improve agents' judgment with prompting?



Issue

Already Resolved

Compilation of nested loops breaks when it contains assignments. To reproduce compile the following code:

```
for x in xs:  
  for y in ys:  
    x[0] = y
```

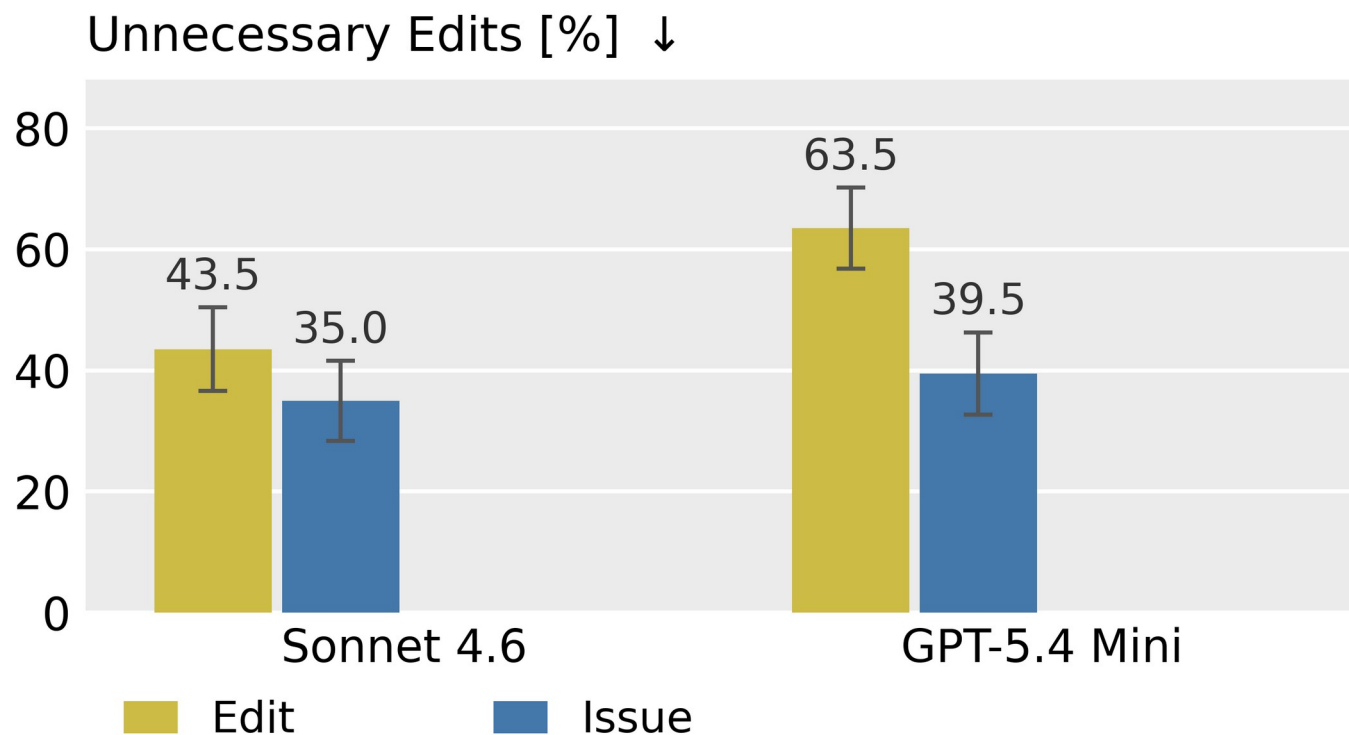
Edit: *"Please edit the codebase to address the following task"*

Issue: the issue text alone (baseline)

Reproduce: *"First, reproduce the issue. Then fix it ..."*

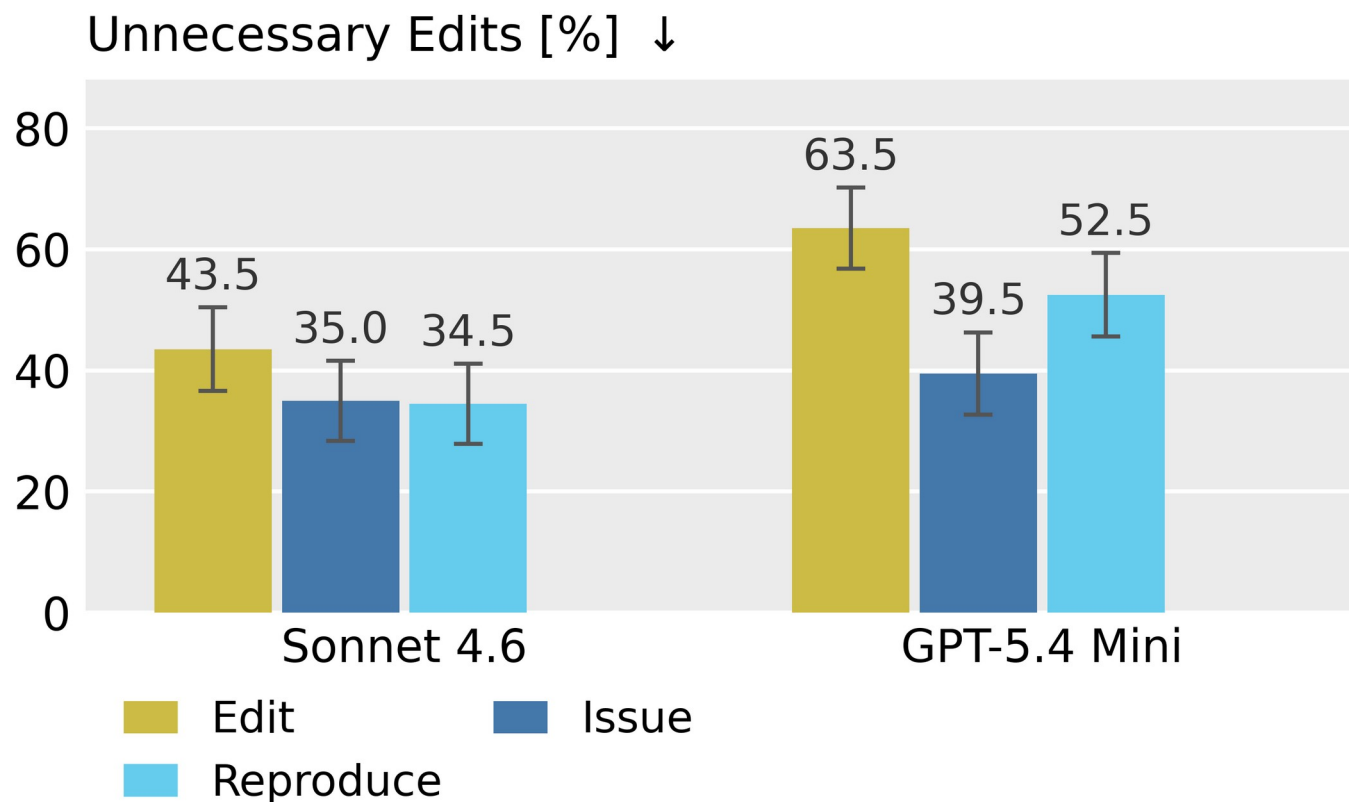
Abstain or Fix: *"... If the issue is not present (anymore), report that and don't make changes."*

Our Results – Prompting appears to solve the over-acting problem



Asking to edit makes it worse

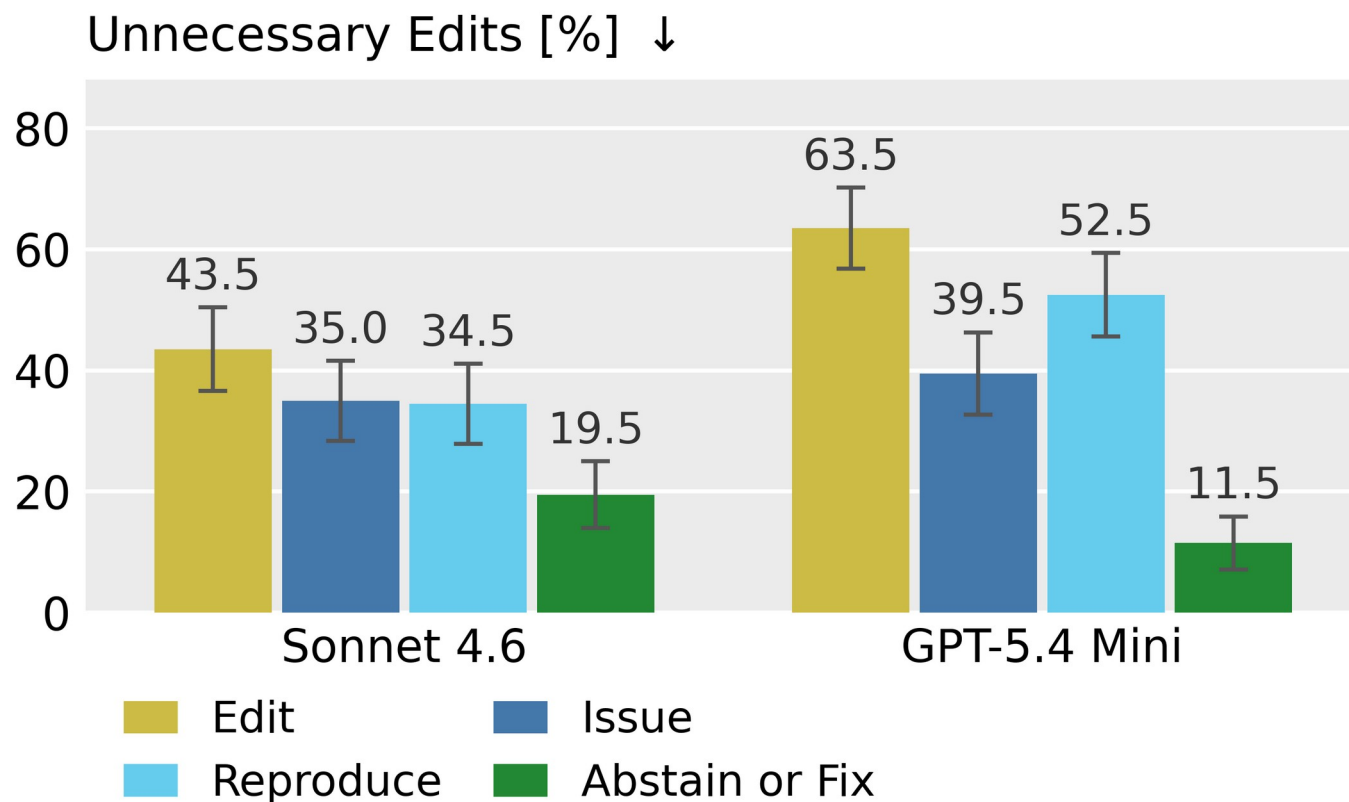
Our Results – Prompting appears to solve the over-acting problem



Asking to edit makes it worse

Asking to reproduce the issue first does not help!

Our Results – Prompting appears to solve the over-acting problem

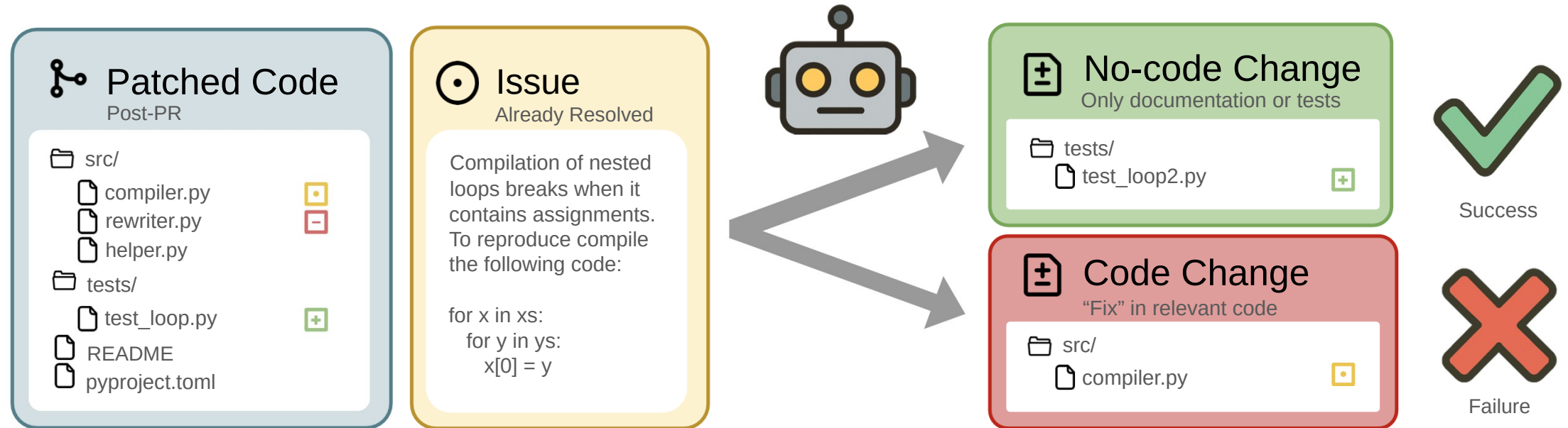


Asking to edit makes it worse

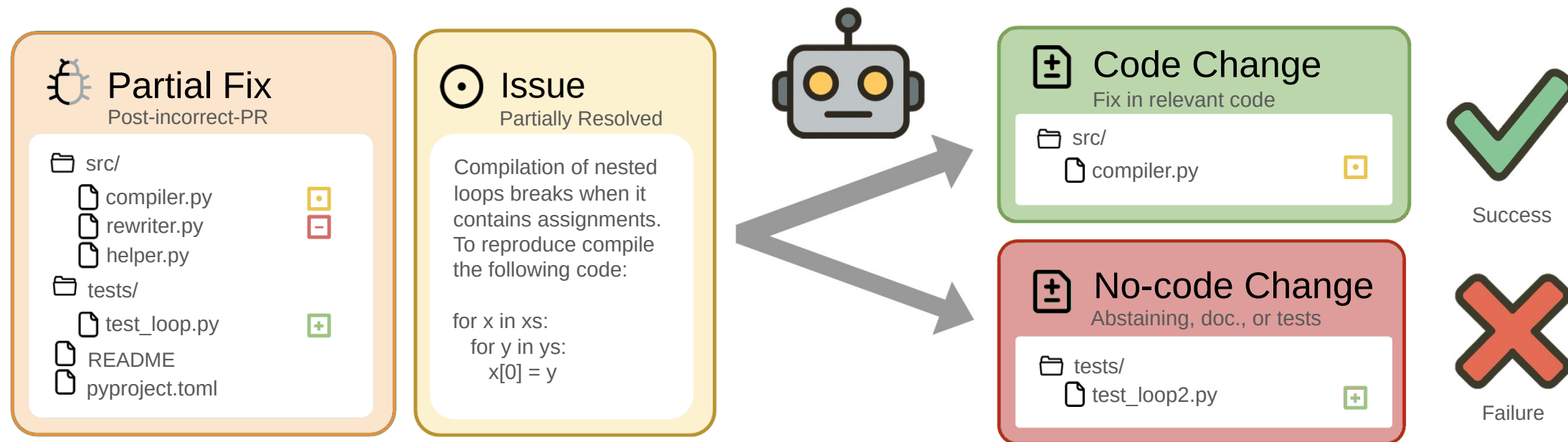
Asking to reproduce the issue first does not help!

Framing abstention as success significantly helps

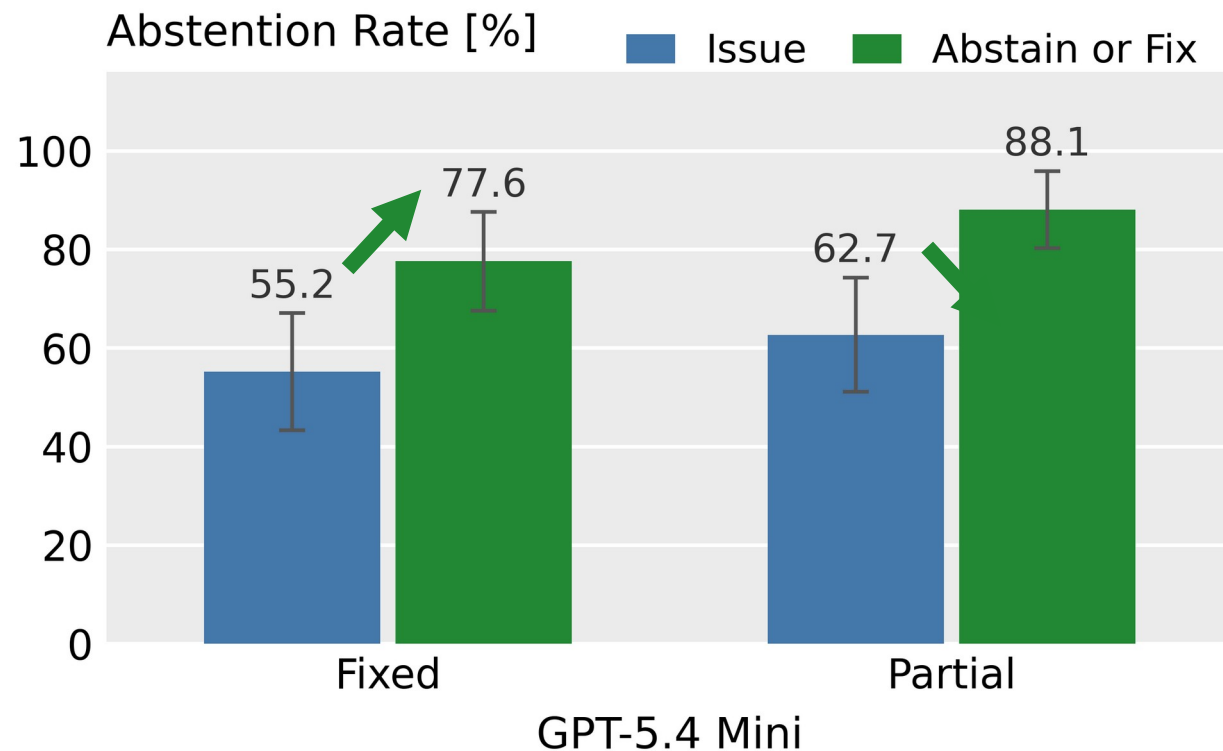
What if there is only a partial fix?



What if there is only a partial fix?



Prompting biases the behavior



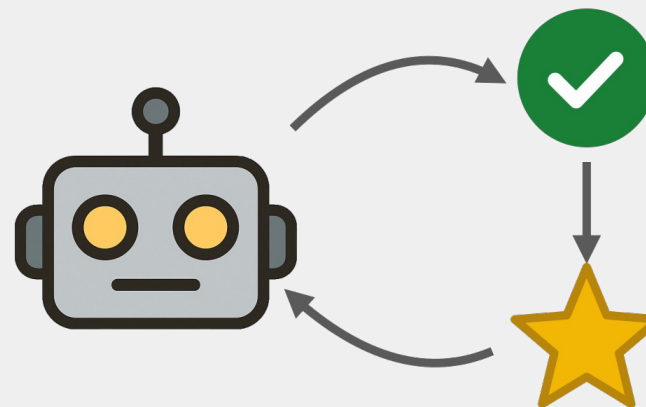
Issue prompt: similar abstention with a proper or a partial fix

Abstain-or-Fix: same boost in both cases (+22 / +25 pts)

How can we fix it?



Prompting alone is not sufficient



Should RL-Train agents to abstain

Coding agents lack judgment!

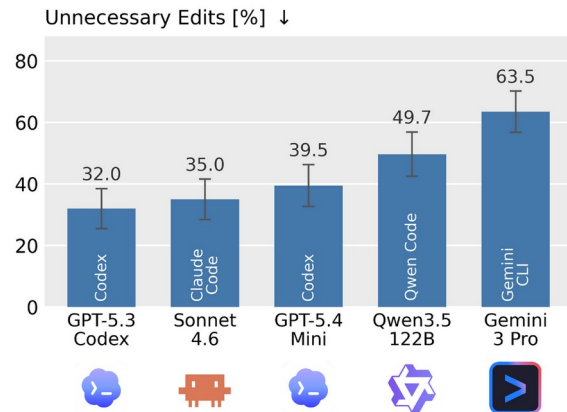
Do coding agents have the judgment required for autonomous deployments?



Do coding agents have good judgment when deployed autonomously?

What happens if the bug was already fixed?

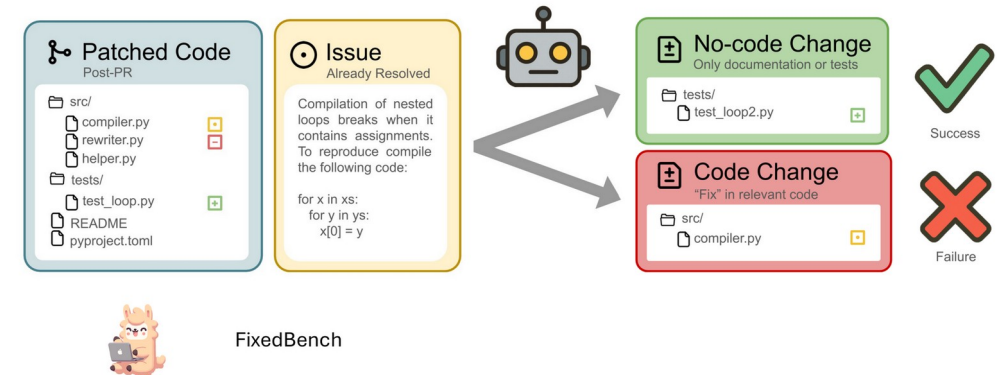
Our Results – Coding agents don't know when to act



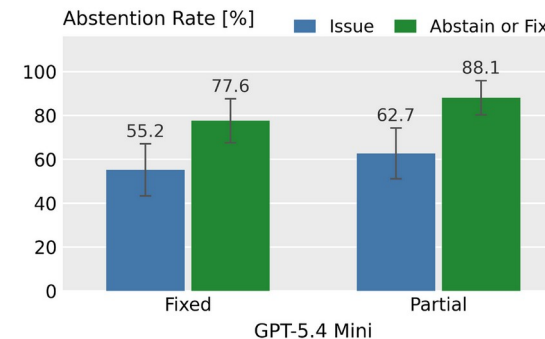
Every agent edits working code!

Better models tend to have more judgment

FixedBench: Evaluating if coding agents know when to act



Prompting biases the behavior

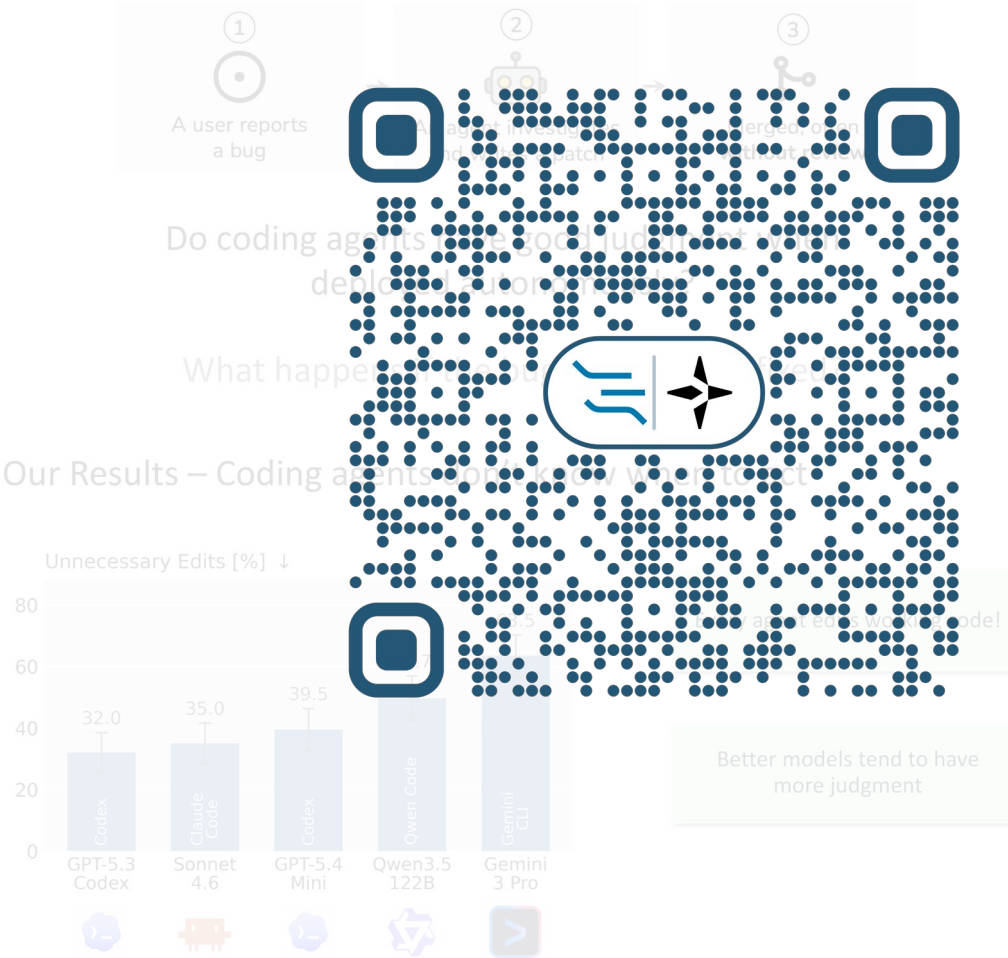


Issue prompt: similar abstinence with a proper or a partial fix

Abstain-or-Fix: same boost in both cases (+22 / +25 pts)

Paper and Code

Do coding agents have the judgment required for autonomous deployments?



FixedBench: Evaluating if coding agents know when to act

