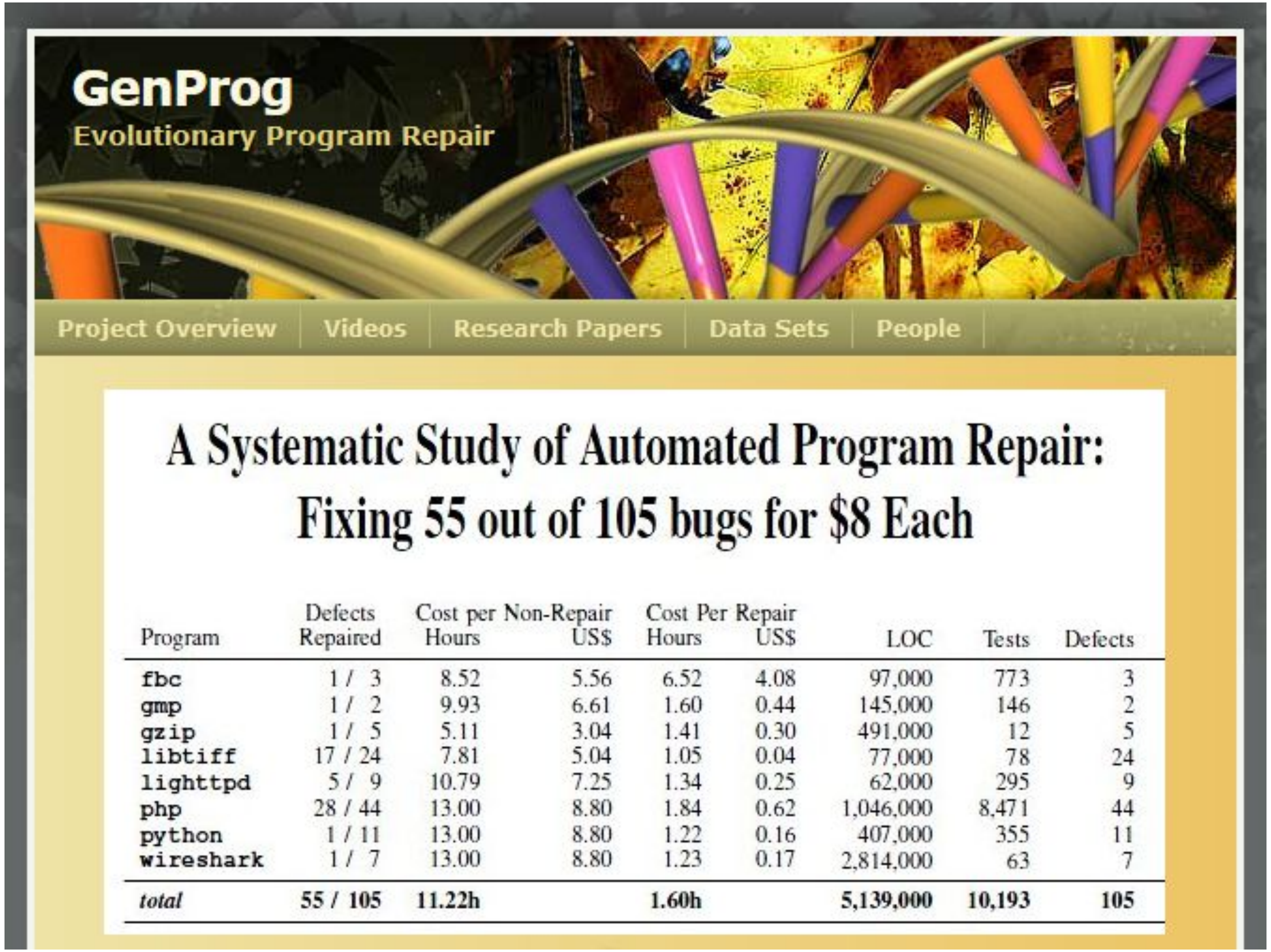
Automated Program Repair

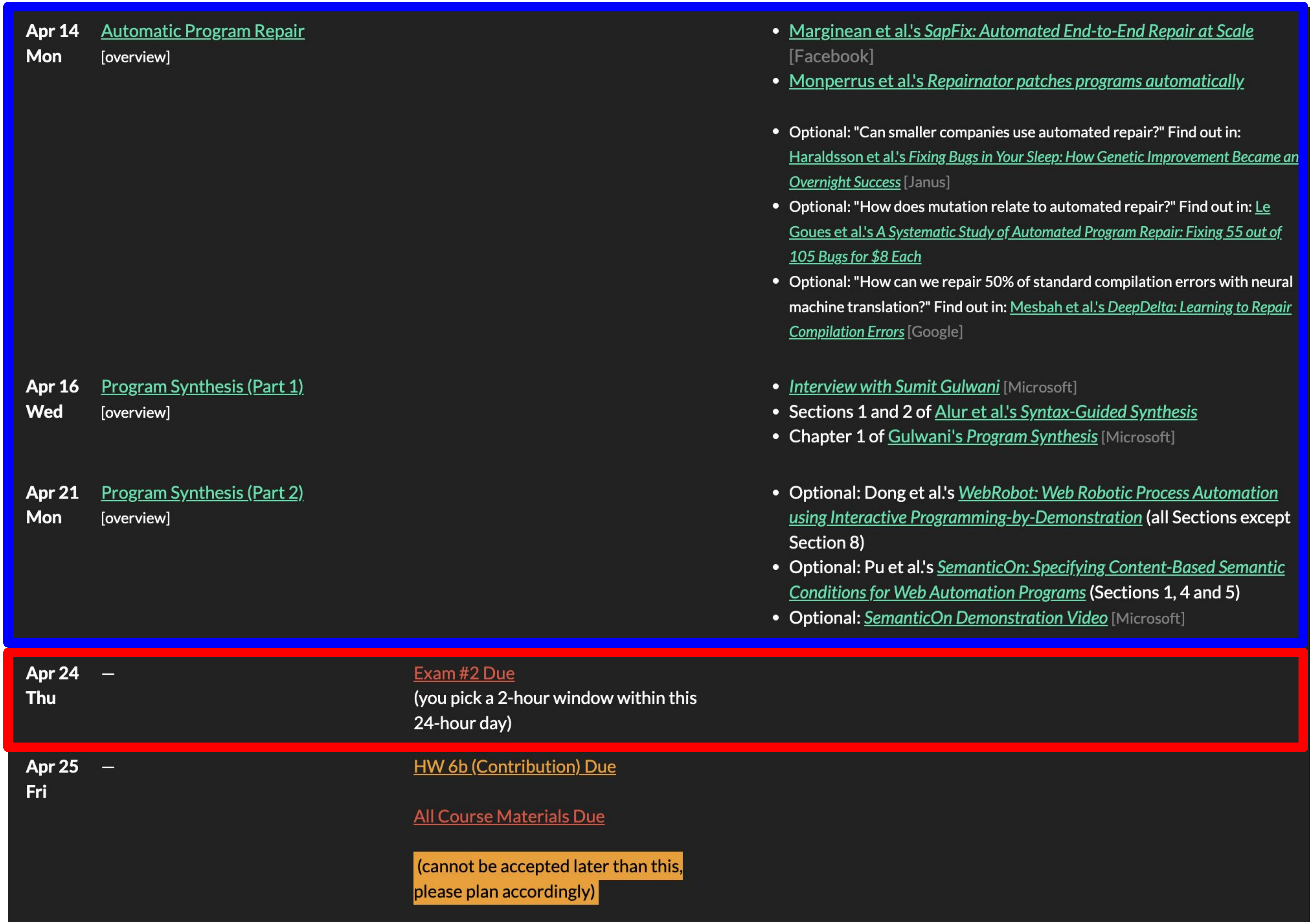


A Systematic Study of Automated Program Repair: Fixing 55 out of 105 bugs for $8 Each

	Defects	Cost per Non-Repair		Cost Per Repair				
Program	Repaired	Hours	US$	Hours	US$	LOC	Tests	Defects
`fbc`	1 / 3	8.52	5.56	6.52	4.08	97,000	773	3
`gmp`	1 / 2	9.93	6.61	1.60	0.44	145,000	146	2
`gzip`	1 / 5	5.11	3.04	1.41	0.30	491,000	12	5
`libtiff`	17 / 24	7.81	5.04	1.05	0.04	77,000	78	24
`lighttpd`	5 / 9	10.79	7.25	1.34	0.25	62,000	295	9
`php`	28 / 44	13.00	8.80	1.84	0.62	1,046,000	8,471	44
`python`	1 / 11	13.00	8.80	1.22	0.16	407,000	355	11
`wireshark`	1 / 7	13.00	8.80	1.23	0.17	2,814,000	63	7
total	**55 / 105**	**11.22h**		**1.60h**		**5,139,000**	**10,193**	**105**

EECS 481
4620

Where Are We?



Apr 14 **Mon**	Automatic Program Repair [overview]		- Marginean et al.'s *SapFix: Automated End-to-End Repair at Scale* [Facebook] - Monperrus et al.'s *Repairnator patches programs automatically* - Optional: "Can smaller companies use automated repair?" Find out in: Haraldsson et al.'s *Fixing Bugs in Your Sleep: How Genetic Improvement Became an Overnight Success* [Janus] - Optional: "How does mutation relate to automated repair?" Find out in: Le Goues et al.'s *A Systematic Study of Automated Program Repair: Fixing 55 out of 105 Bugs for $8 Each* - Optional: "How can we repair 50% of standard compilation errors with neural machine translation?" Find out in: Mesbah et al.'s *DeepDelta: Learning to Repair Compilation Errors* [Google]
Apr 16 **Wed**	Program Synthesis (Part 1) [overview]		- *Interview with Sumit Gulwani* [Microsoft] - Sections 1 and 2 of Alur et al.'s *Syntax-Guided Synthesis* - Chapter 1 of Gulwani's *Program Synthesis* [Microsoft]
Apr 21 **Mon**	Program Synthesis (Part 2) [overview]		- Optional: Dong et al.'s *WebRobot: Web Robotic Process Automation using Interactive Programming-by-Demonstration* (all Sections except Section 8) - Optional: Pu et al.'s *SemanticOn: Specifying Content-Based Semantic Conditions for Web Automation Programs* (Sections 1, 4 and 5) - Optional: *SemanticOn Demonstration Video* [Microsoft]
Apr 24 **Thu**	—	Exam #2 Due (you pick a 2-hour window within this 24-hour day)	
Apr 25 **Fri**	—	HW 6b (Contribution) Due All Course Materials Due (cannot be accepted later than this, please plan accordingly)	

The Never-Ending Story

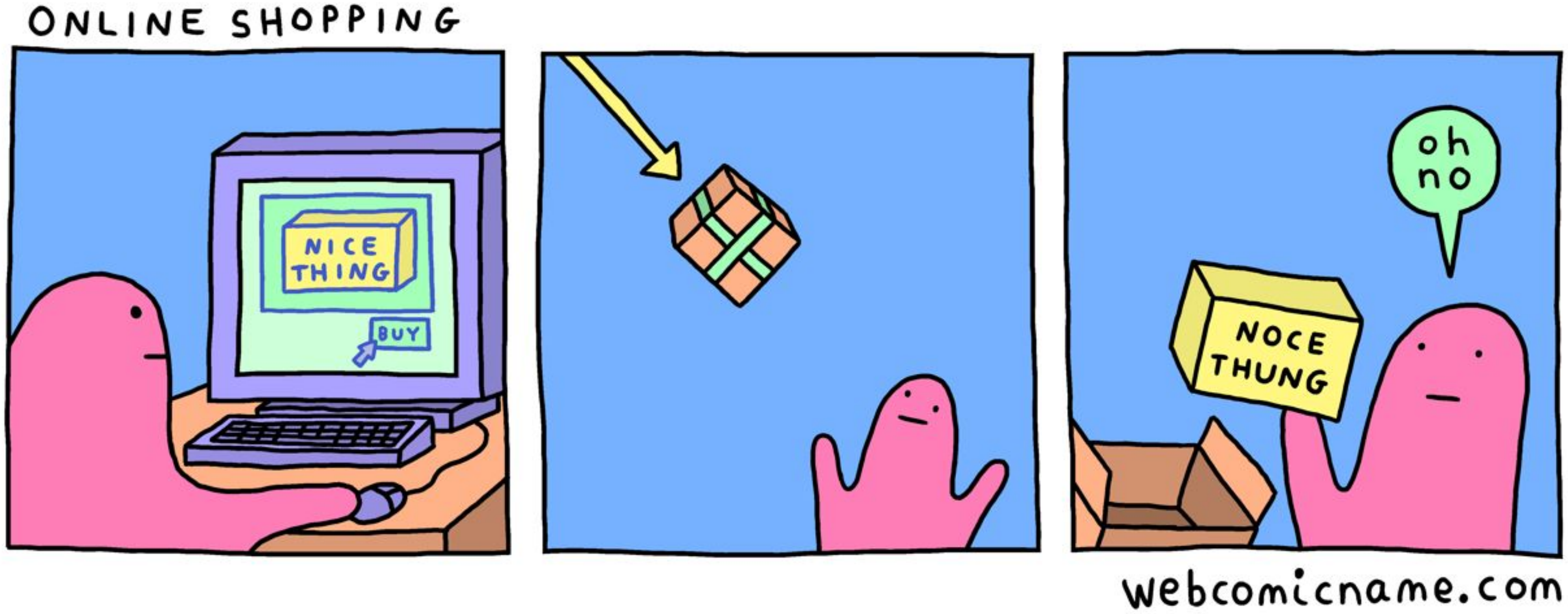
- Today we will use recent advances in **automated program repair** to touch on all of the **lecture topics** from this course

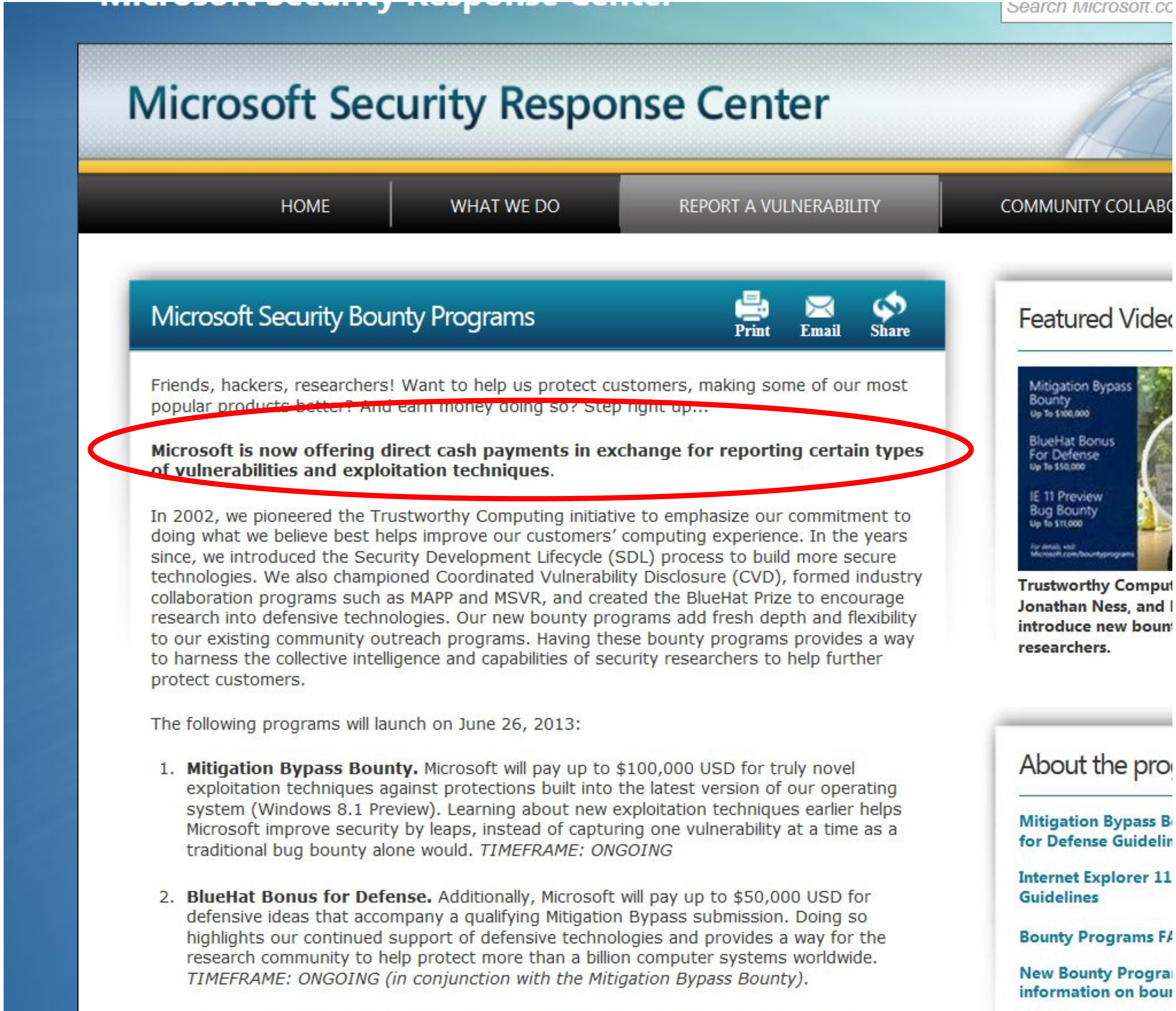
Leading Question

- How do software companies find/fix bugs?

Speculative Fiction

- What if large, trusted companies paid **strangers** online to find and fix their normal and critical bugs?





Microsoft Security Bounty Programs

Friends, hackers, researchers! Want to help us protect customers, making some of our most popular products better? And earn money doing so? Step right up...

Microsoft is now offering direct cash payments in exchange for reporting certain types of vulnerabilities and exploitation techniques.

In 2002, we pioneered the Trustworthy Computing initiative to emphasize our commitment to doing what we believe best helps improve our customers' computing experience. In the years since, we introduced the Security Development Lifecycle (SDL) process to build more secure technologies. We also championed Coordinated Vulnerability Disclosure (CVD), formed industry collaboration programs such as MAPP and MSVR, and created the BlueHat Prize to encourage research into defensive technologies. Our new bounty programs add fresh depth and flexibility to our existing community outreach programs. Having these bounty programs provides a way to harness the collective intelligence and capabilities of security researchers to help further protect customers.

The following programs will launch on June 26, 2013:

1. **Mitigation Bypass Bounty.** Microsoft will pay up to $100,000 USD for truly novel exploitation techniques against protections built into the latest version of our operating system (Windows 8.1 Preview). Learning about new exploitation techniques earlier helps Microsoft improve security by leaps, instead of capturing one vulnerability at a time as a traditional bug bounty alone would. *TIMEFRAME: ONGOING*

2. **BlueHat Bonus for Defense.** Additionally, Microsoft will pay up to $50,000 USD for defensive ideas that accompany a qualifying Mitigation Bypass submission. Doing so highlights our continued support of defensive technologies and provides a way for the research community to help protect more than a billion computer systems worldwide. *TIMEFRAME: ONGOING (in conjunction with the Mitigation Bypass Bounty).*

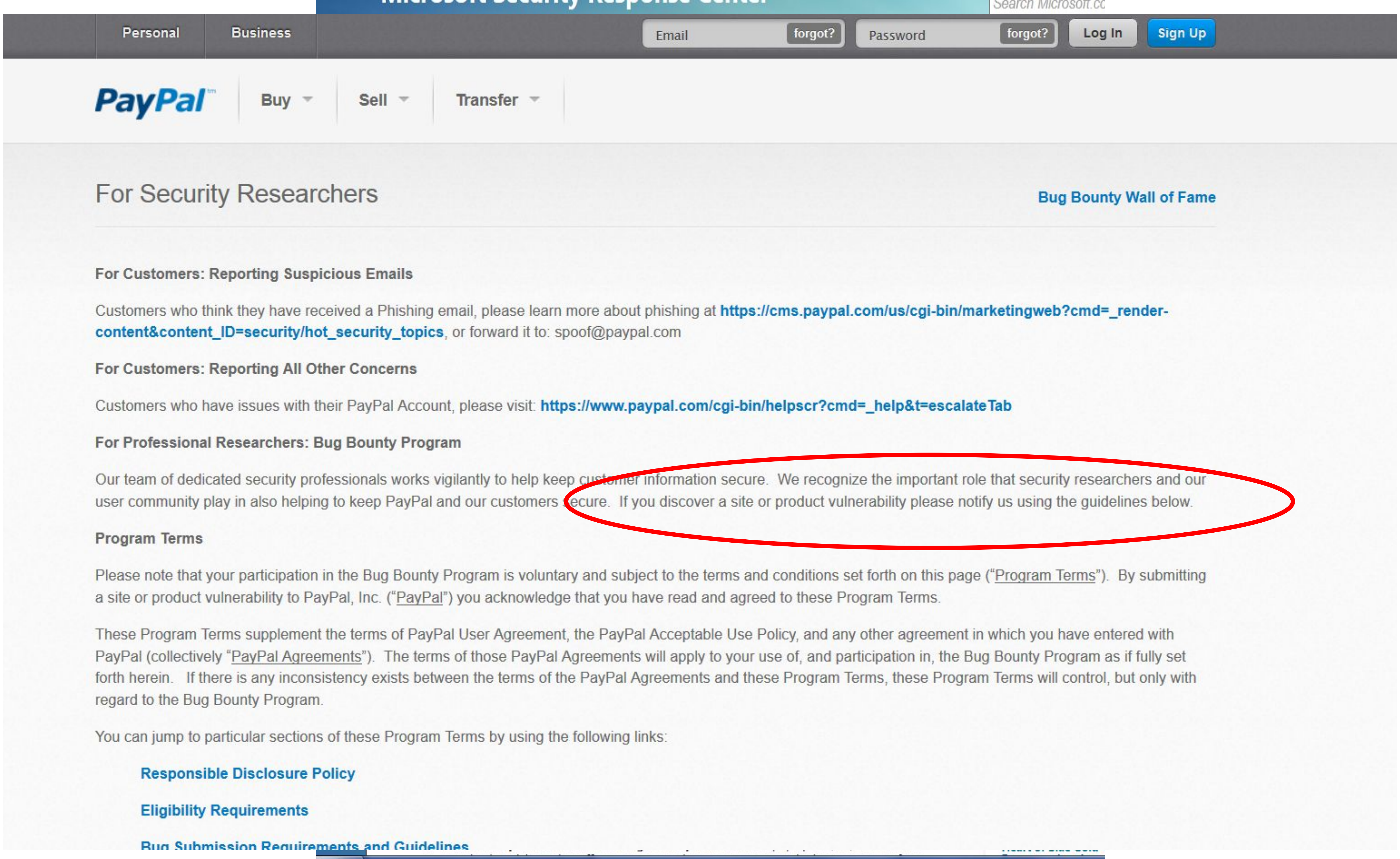
Trustworthy Comput
Jonathan Ness, and I
introduce new bount
researchers.

Mitigation Bypass B
for Defense Guidelin

Internet Explorer 11
Guidelines

Bounty Programs FA

New Bounty Progra
information on bour



For Security Researchers

Bug Bounty Wall of Fame

For Customers: Reporting Suspicious Emails

Customers who think they have received a Phishing email, please learn more about phishing at **https://cms.paypal.com/us/cgi-bin/marketingweb?cmd=_render-content&content_ID=security/hot_security_topics**, or forward it to: spoof@paypal.com

For Customers: Reporting All Other Concerns

Customers who have issues with their PayPal Account, please visit: **https://www.paypal.com/cgi-bin/helpscr?cmd=_help&t=escalateTab**

For Professional Researchers: Bug Bounty Program

Our team of dedicated security professionals works vigilantly to help keep customer information secure. We recognize the important role that security researchers and our user community play in also helping to keep PayPal and our customers secure. If you discover a site or product vulnerability please notify us using the guidelines below.

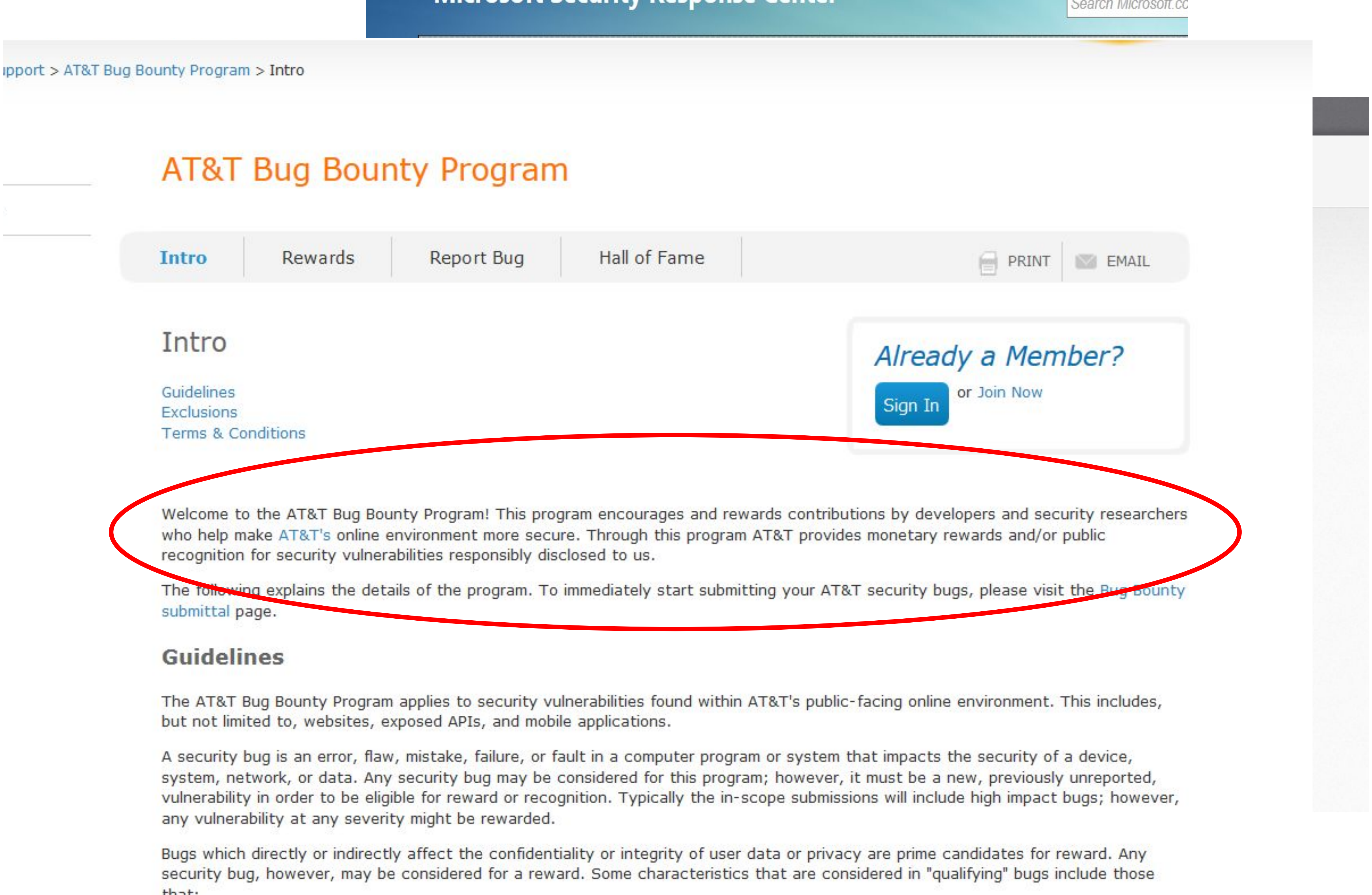
Program Terms

Please note that your participation in the Bug Bounty Program is voluntary and subject to the terms and conditions set forth on this page ("Program Terms"). By submitting a site or product vulnerability to PayPal, Inc. ("PayPal") you acknowledge that you have read and agreed to these Program Terms.

These Program Terms supplement the terms of PayPal User Agreement, the PayPal Acceptable Use Policy, and any other agreement in which you have entered with PayPal (collectively "PayPal Agreements"). The terms of those PayPal Agreements will apply to your use of, and participation in, the Bug Bounty Program as if fully set forth herein. If there is any inconsistency exists between the terms of the PayPal Agreements and these Program Terms, these Program Terms will control, but only with regard to the Bug Bounty Program.

You can jump to particular sections of these Program Terms by using the following links:

- **Responsible Disclosure Policy**
- **Eligibility Requirements**
- **Bug Submission Requirements and Guidelines**



AT&T Bug Bounty Program

Intro

Guidelines
Exclusions
Terms & Conditions

Welcome to the AT&T Bug Bounty Program! This program encourages and rewards contributions by developers and security researchers who help make AT&T's online environment more secure. Through this program AT&T provides monetary rewards and/or public recognition for security vulnerabilities responsibly disclosed to us.

The following explains the details of the program. To immediately start submitting your AT&T security bugs, please visit the Bug Bounty submittal page.

Guidelines

The AT&T Bug Bounty Program applies to security vulnerabilities found within AT&T's public-facing online environment. This includes, but not limited to, websites, exposed APIs, and mobile applications.

A security bug is an error, flaw, mistake, failure, or fault in a computer program or system that impacts the security of a device, system, network, or data. Any security bug may be considered for this program; however, it must be a new, previously unreported, vulnerability in order to be eligible for reward or recognition. Typically the in-scope submissions will include high impact bugs; however, any vulnerability at any severity might be rewarded.

Bugs which directly or indirectly affect the confidentiality or integrity of user data or privacy are prime candidates for reward. Any security bug, however, may be considered for a reward. Some characteristics that are considered in "qualifying" bugs include those that:

Microsoft Security Response Center

Search Microsoft.co

For Security Researchers

For Customers: Reporting Suspicious Emails

Customers who think they have received

content&content_ID=security/hot_secu

For Customers: Reporting All Other C

Customers who have issues with their Pa

For Professional Researchers: Bug Bo

Our team of dedicated security professior user community play in also helping to ke

Program Terms

Please note that your participation in the a site or product vulnerability to PayPal, I

These Program Terms supplement the te PayPal (collectively "PayPal Agreements" forth herein. If there is any inconsistency regard to the Bug Bounty Program.

You can jump to particular sections of the

Responsible Disclosure Policy

Eligibility Requirements

Bug Submission Requirements and Guidelines

ıpport > AT&T Bug Bounty Program > Intro

AT&T Bug Bounty Program

elopers and security researchers rewards and/or public

ugs, please visit the Bug Bounty

e environment. This includes,

the security of a device, new, previously unreported, ıde high impact bugs; however, any vulnerability at any severity might be rewarded.

Bugs which directly or indirectly affect the confidentiality or integrity of user data or privacy are prime candidates for reward. Any security bug, however, may be considered for a reward. Some characteristics that are considered in "qualifying" bugs include those that:

(Raise Your Hand If True)

I have used software produced by Microsoft, PayPal, AT&T, Facebook, Mozilla, Google *or* Youtube.

Bug Bounties

- If you trust your triage and code review processes, **anyone** can submit a candidate bug report or candidate patch
- **Bug Bounties** combine **defect reporting and triage** with **pass-around code review**
- Finding, fixing and ignoring bugs are all so expensive that it is now (~2013+) economical to pay untrusted strangers to submit candidate defect reports and patches

Bug Bounties and Large Companies

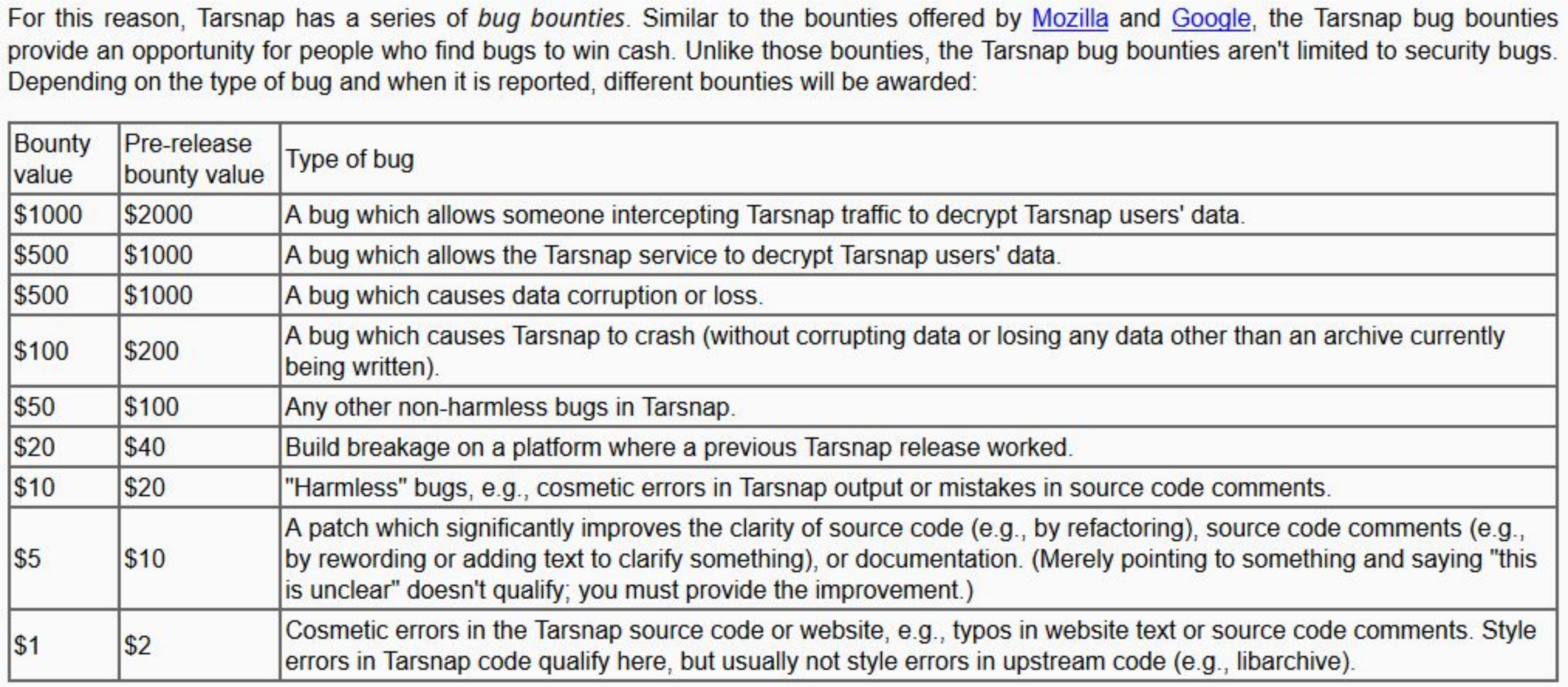
- “We get hundreds of reports every day. Many of our best reports come from people whose English isn't great – though this can be challenging, it's something we work with just fine and we have paid out over $1 million to hundreds of reporters.”

  – Matt Jones, Meta/Facebook Software Engineering

Bug Bounties and Small Companies

- Only 38% of the submissions were true positives (harmless, minor or major): “Worth the money? Every penny.” - Colin Percival, Tarsnap

Tarsnap
Online backups for the truly paranoid

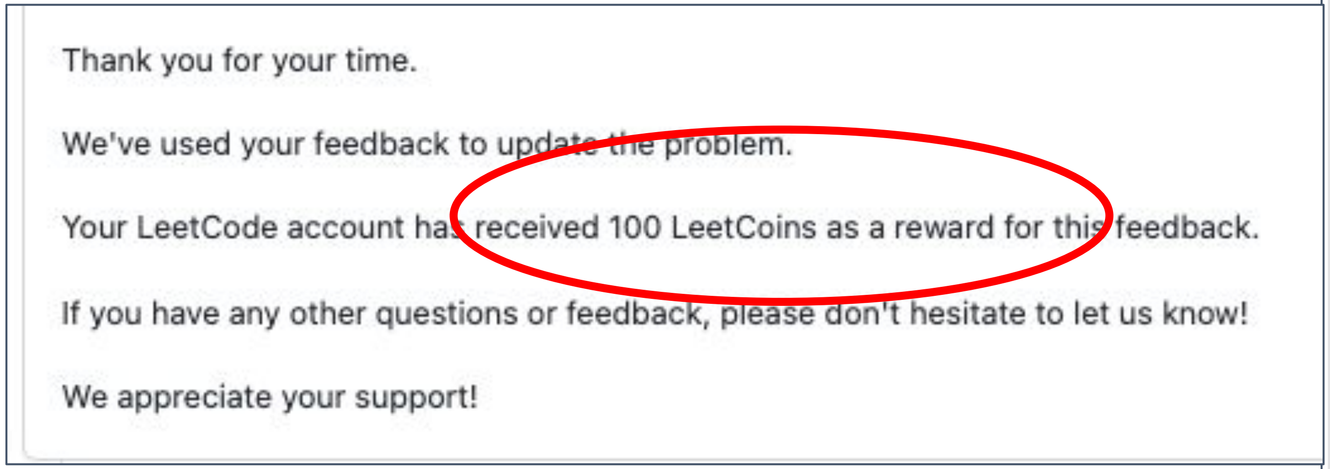


For this reason, Tarsnap has a series of *bug bounties*. Similar to the bounties offered by Mozilla and Google, the Tarsnap bug bounties provide an opportunity for people who find bugs to win cash. Unlike those bounties, the Tarsnap bug bounties aren't limited to security bugs. Depending on the type of bug and when it is reported, different bounties will be awarded:

Bounty value	Pre-release bounty value	Type of bug
$1000	$2000	A bug which allows someone intercepting Tarsnap traffic to decrypt Tarsnap users' data.
$500	$1000	A bug which allows the Tarsnap service to decrypt Tarsnap users' data.
$500	$1000	A bug which causes data corruption or loss.
$100	$200	A bug which causes Tarsnap to crash (without corrupting data or losing any data other than an archive currently being written).
$50	$100	Any other non-harmless bugs in Tarsnap.
$20	$40	Build breakage on a platform where a previous Tarsnap release worked.
$10	$20	"Harmless" bugs, e.g., cosmetic errors in Tarsnap output or mistakes in source code comments.
$5	$10	A patch which significantly improves the clarity of source code (e.g., by refactoring), source code comments (e.g., by rewording or adding text to clarify something), or documentation. (Merely pointing to something and saying "this is unclear" doesn't qualify; you must provide the improvement.)
$1	$2	Cosmetic errors in the Tarsnap source code or website, e.g., typos in website text or source code comments. Style errors in Tarsnap code qualify here, but usually not style errors in upstream code (e.g., libarchive).

LeetCode Example

- Report “missing test cases” on LeetCode
- Rewards don’t have to be Cash!



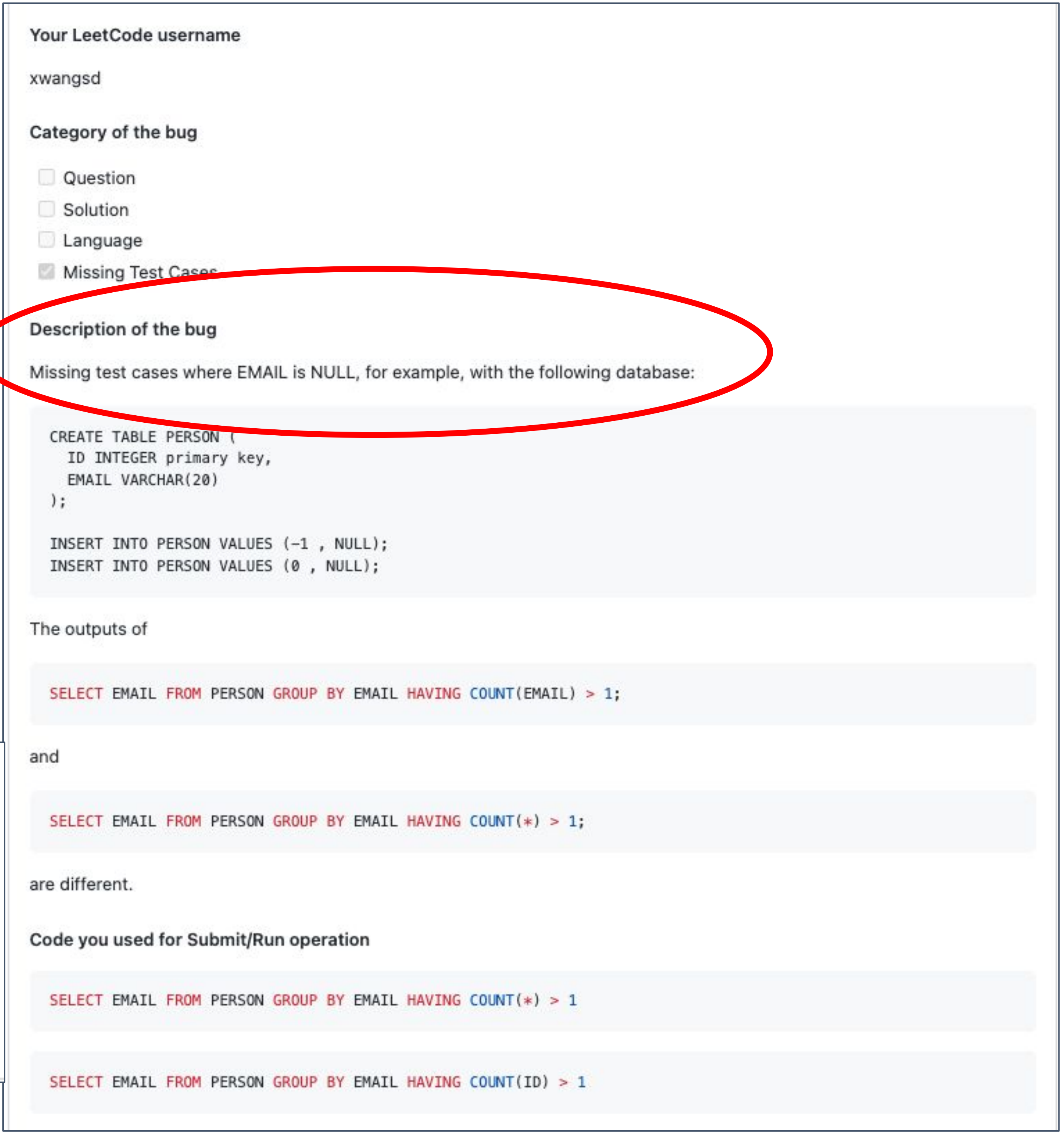
Thank you for your time.

We've used your feedback to update the problem.

Your LeetCode account has received 100 LeetCoins as a reward for this feedback.

If you have any other questions or feedback, please don't hesitate to let us know!

We appreciate your support!



Your LeetCode username

xwangsd

Category of the bug

□ Question
□ Solution
□ Language
☑ Missing Test Cases

Description of the bug

Missing test cases where EMAIL is NULL, for example, with the following database:

```
CREATE TABLE PERSON (
  ID INTEGER primary key,
  EMAIL VARCHAR(20)
);

INSERT INTO PERSON VALUES (-1 , NULL);
INSERT INTO PERSON VALUES (0 , NULL);
```

The outputs of

```
SELECT EMAIL FROM PERSON GROUP BY EMAIL HAVING COUNT(EMAIL) > 1;
```

and

```
SELECT EMAIL FROM PERSON GROUP BY EMAIL HAVING COUNT(*) > 1;
```

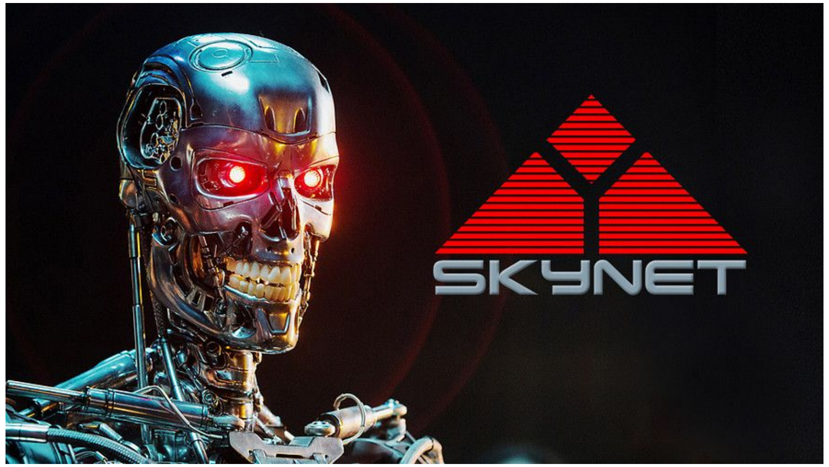
are different.

Code you used for Submit/Run operation

```
SELECT EMAIL FROM PERSON GROUP BY EMAIL HAVING COUNT(*) > 1
```

```
SELECT EMAIL FROM PERSON GROUP BY EMAIL HAVING COUNT(ID) > 1
```

A Modest Proposal

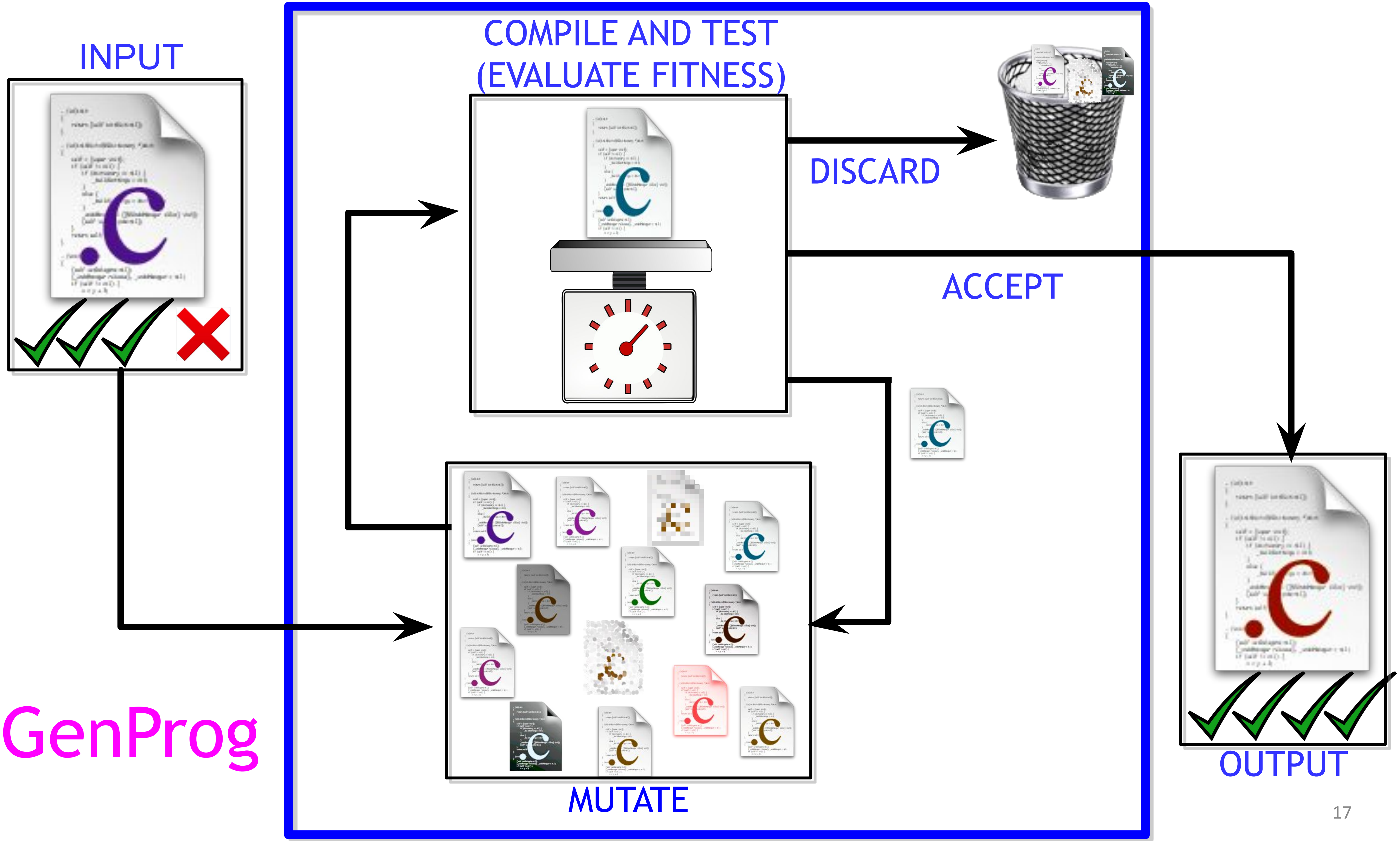


- Using techniques from this class
- We can **automatically** find and fix defects
  - Rather than, or in addition to, paying strangers
- Given a program …
  - Source code, binary code, etc.
- … and evidence of a bug …
  - Passing and failing tests, crashes, etc.
- … fix that bug.
  - Create a textual patch (pull request)

How could this possibly work?

- Many faults can be localized to a small area
  - Even if your program is a million lines of code, **fault localization** can narrow it to 10-100 lines
- Many defects can be fixed with small changes
  - **Mutation** (test metrics) can generate candidate patches from simple edits
  - A **search-based software engineering** problem
- Can use regression **testing** (inputs and oracles, continuous integration) to assess patch quality

[Weimer et al. *Automatically Finding Patches Using Genetic Programming*. Best Paper Award. IFIP TC2 Manfred Paul Award. SIGEVO “Humies” Gold Award. Ten-Year Impact Award.]



INPUT

COMPILE AND TEST
(EVALUATE FITNESS)

DISCARD

ACCEPT

MUTATE

OUTPUT

GenProg

Does this really work?

Name	Subjects	Tests	Bugs	Notes
AFix	2 Mloc	–	8	Concurrency, guarantees
ARC	–	–	–	Concurrency, SBSE
ARMOR	6 progs.	–	3 + –	Identifies workarounds
Axis	13 progs.	–	–	Concurrency, guarantees, Petri nets
AutoFix-E	21 Kloc	650	42	Contracts, guarantees
CASC	1 Kloc	–	5	Co-evolves tests and programs
ClearView	Firefox	57	9	Red Team quality evaluation
Coker Hafiz	15 Mloc	–	7 / –	Integer bugs only, guarantees
Debroy Wong	76 Kloc	22,500	135	Mutation, fault localization focus
Demsky et al.	3 progs.	–	–	Data struct consistency, Red Team
FINCH	13 tasks	–	–	Evolves unrestricted bytecode
GenProg	5 Mloc	10,000	105	Human-competitive, SBSE
Gopinath et al.	2 methods.	–	20	Heap specs, SAT
Jolt	5 progs.	–	8	Escape infinite loops at run-time
Juzi	7 progs.	–	20 + –	Data struct consistency, models
PACHIKA	110 Kloc	2,700	26	Differences in behavior models
PAR	480 Kloc	25,000	119	Human-based patches, quality study
SemFix	12 Kloc	250	90	Symex, constraints, synthesis
Sidiroglou et al.	17 progs.	–	17	Buffer overflows

Scales well

Name	Subjects	Tests	Bugs	Notes
AFix	2 Mloc	–	8	Concurrency, guarantees
ARC	–	–	–	Concurrency, SBSE
ARMOR	6 progs.	–	3 + –	Identifies workarounds
Axis	13 progs.	–	–	Concurrency, guarantees, Petri nets
AutoFix-E	21 Kloc	650	42	Contracts, guarantees
CASC	1 Kloc	–	5	Co-evolves tests and programs
ClearView	Firefox	57	9	Red Team quality evaluation
Coker Hafiz	15 Mloc	–	7 / –	Integer bugs only, guarantees
Debroy Wong	76 Kloc	22,500	135	Mutation, fault localization focus
Demsky et al.	3 progs.	–	–	Data struct consistency, Red Team
FINCH	13 tasks	–	–	Evolves unrestricted bytecode
GenProg	5 Mloc	10,000	105	Human-competitive, SBSE
Gopinath et al.	2 methods.	–	20	Heap specs, SAT
Jolt	5 progs.	–	8	Escape infinite loops at run-time
Juzi	7 progs.	–	20 + –	Data struct consistency, models
PACHIKA	110 Kloc	2,700	26	Differences in behavior models
PAR	480 Kloc	25,000	119	Human-based patches, quality study
SemFix	12 Kloc	250	90	Symex, constraints, synthesis
Sidiroglou et al.	17 progs.	–	17	Buffer overflows

Can fix bugs

Name	Subjects	Tests	Bugs	Notes
AFix	2 Mloc	–	8	Concurrency, guarantees
ARC	–	–	–	Concurrency, SBSE
ARMOR	6 progs.	–	3 + –	Identifies workarounds
Axis	13 progs.	–	–	Concurrency, guarantees, Petri nets
AutoFix-E	21 Kloc	650	42	Contracts, guarantees
CASC	1 Kloc	–	5	Co-evolves tests and programs
ClearView	Firefox	57	9	Red Team quality evaluation
Coker Hafiz	15 Mloc	–	7 / –	Integer bugs only, guarantees
Debroy Wong	76 Kloc	22,500	135	Mutation, fault localization focus
Demsky et al.	3 progs.	–	–	Data struct consistency, Red Team
FINCH	13 tasks	–	–	Evolves unrestricted bytecode
GenProg	5 Mloc	10,000	105	Human-competitive, SBSE
Gopinath et al.	2 methods.	–	20	Heap specs, SAT
Jolt	5 progs.	–	8	Escape infinite loops at run-time
Juzi	7 progs.	–	20 + –	Data struct consistency, models
PACHIKA	110 Kloc	2,700	26	Differences in behavior models
PAR	480 Kloc	25,000	119	Human-based patches, quality study
SemFix	12 Kloc	250	90	Symex, constraints, synthesis
Sidiroglou et al.	17 progs.	–	17	Buffer overflows

Can provide guarantee

Name	Subjects	Tests	Bugs	Notes
AFix	2 Mloc	–	8	Concurrency, guarantees
ARC	–	–	–	Concurrency, SBSE
ARMOR	6 progs.	–	3 + –	Identifies workarounds
Axis	13 progs.	–	–	Concurrency, guarantees, Petri nets
AutoFix-E	21 Kloc	650	42	Contracts, guarantees
CASC	1 Kloc	–	5	Co-evolves tests and programs
ClearView	Firefox	57	9	Red Team quality evaluation
Coker Hafiz	15 Mloc	–	7 / –	Integer bugs only, guarantees
Debroy Wong	76 Kloc	22,500	135	Mutation, fault localization focus
Demsky et al.	3 progs.	–	–	Data struct consistency, Red Team
FINCH	13 tasks	–	–	Evolves unrestricted bytecode
GenProg	5 Mloc	10,000	105	Human-competitive, SBSE
Gopinath et al.	2 methods.	–	20	Heap specs, SAT
Jolt	5 progs.	–	8	Escape infinite loops at run-time
Juzi	7 progs.	–	20 + –	Data struct consistency, models
PACHIKA	110 Kloc	2,700	26	Differences in behavior models
PAR	480 Kloc	25,000	119	Human-based patches, quality study
SemFix	12 Kloc	250	90	Symex, constraints, synthesis
Sidiroglou et al.	17 progs.	–	17	Buffer overflows

Consideration: Minimizing Patches

- A GenProg patch may contain extraneous/redundant edits
  - Add “close();” vs. add “close(); x = x + 0;”
  - Both pass all tests, but …
- Longer patches are harder to read
- Extraneous edits may only appear safe because of weak test suites: avoid unneeded code churn
- How to minimize? After the repair search, use **delta debugging** (hypothesis testing) to find a passing 1-minimal edit subset

Consideration: Minimizing Costs (time, ..)

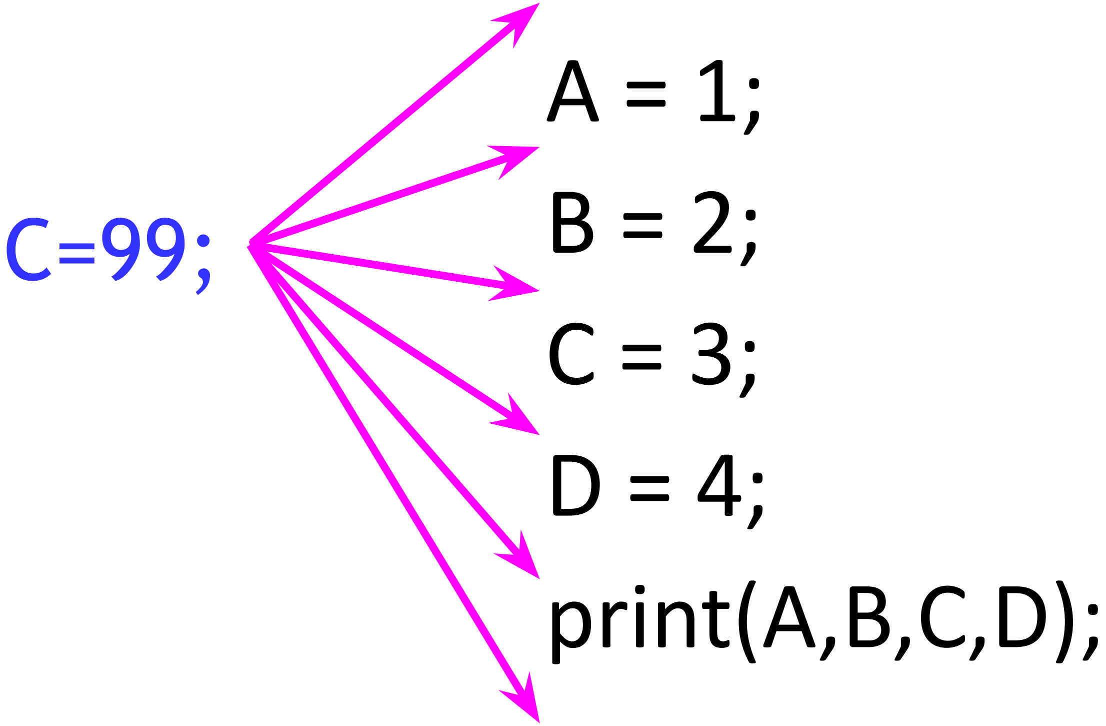
- Can stop generating candidate mutants when a valid repair is found, parallelize in the cloud

  [Le Goues et al. *A Systematic Study of Automated Program Repair: Fixing 55 out of 105 bugs for $8 Each.*]

- Each repair must pass the entire test suite
  - Running tests is the dominant cost of automated program repair
  - Use test suite prioritization and minimization
  - Stop evaluating as soon as a single test fails
    - Even one failure → Not a valid repair!

Can We Avoid Testing In The First Place? (An even better way to minimize cost..)

- If P1 and P2 are semantically equivalent they must have the same functional test behavior
- Consider this insertion:



C=99;

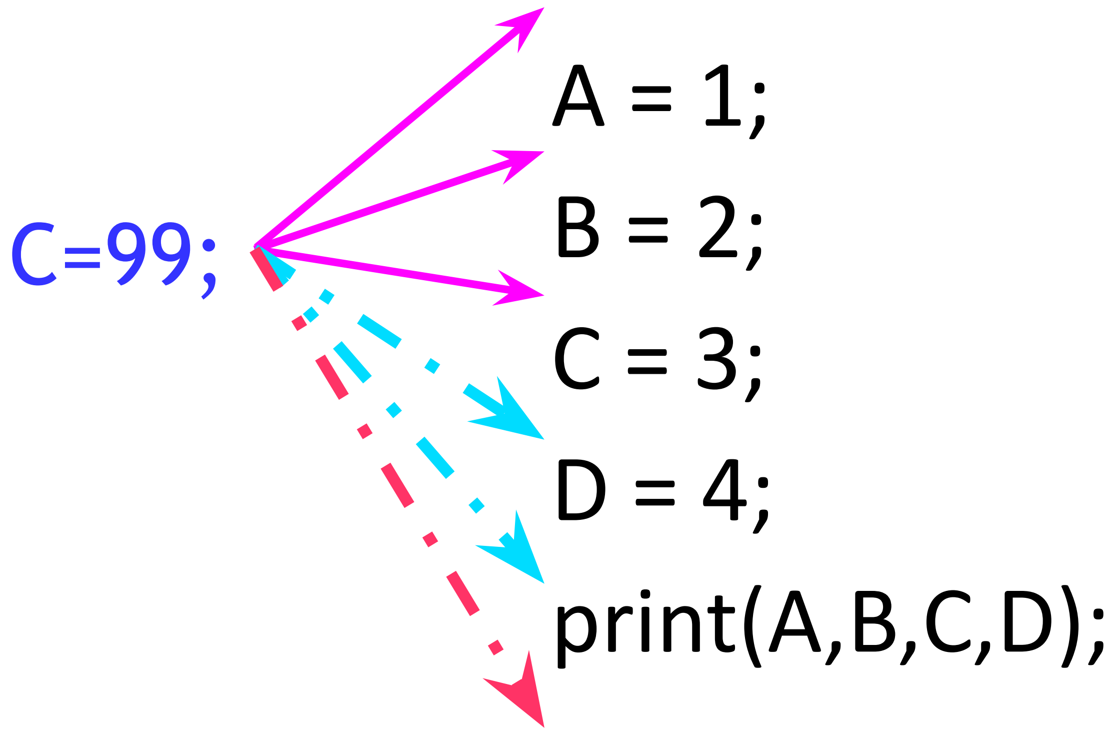
```
A = 1;
B = 2;
C = 3;
D = 4;
print(A,B,C,D);
```

Quiz:

Among six mutants, which one(s) can be avoided?

Can We Avoid Testing In The First Place? (An even better way to minimize cost..)

- If P1 and P2 are semantically equivalent they must have the same functional test behavior
- Consider this insertion:



C=99;

```
A = 1;
B = 2;
C = 3;
D = 4;
print(A,B,C,D);
```

Static Analysis

- If we had a cheap way to approximately decide if two programs are equivalent
  - We wouldn't need to test any candidate patch that is equivalent to a previously-tested patch
  - (Cluster or quotient the search space into equivalence classes with respect to this relation)
- We use **static analysis** (like a dataflow analysis for dead code or constant propagation) to decide this: 10x reduction in search space

[Weimer et al. *Leveraging Program Equivalence for Adaptive Program Repair: Models and First Results.*]

Consideration: Design Patterns

- In mutation testing, the mutation operators are based on common human mistakes
- In program repair, use human edits (likely to be correct) or **design patterns**
  - “Add a null check” or “Use a singleton pattern”
- Mine 60,000 human-written patches (e.g., from github) to learn the 10 most common fix templates
  - Resulting approach fixes 70% more bugs
  - Human study of non-student developers (n=68): such patches are 20% more acceptable

[Kim et al. *Automatic Patch Generation Learned from Human-Written Patches.* Best paper award.]

ChatGPT and Large Language Models

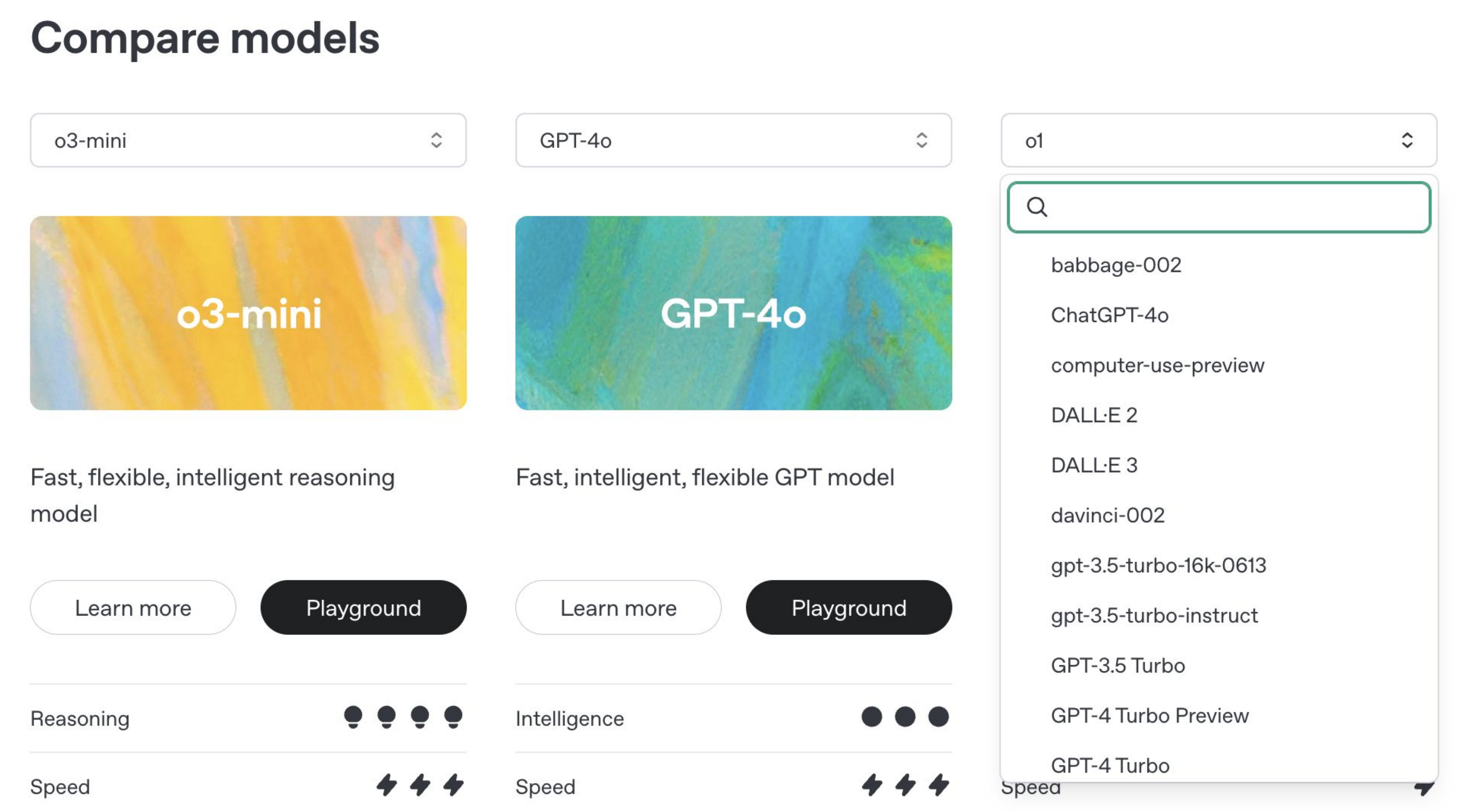
- Which company/university developed ChatGPT?
  - ○ MIT
  - ○ Stanford
  - ○ Microsoft
  - ○ Google
  - ○ OpenAI

ChatGPT and Large Language Models

- Which company/university developed ChatGPT?
  - MIT
  - Stanford
  - Microsoft
  - Google
  - **OpenAI**

ChatGPT and Large Language Models

- GPT = Generative Pre-trained Transformers
  - … are a family of (large) language models trained on a large corpus of **text** data
- ChatGPT (Nov 2022 release) was based on GPT-3.5
- March 14 2023: GPT-4 release
- **Now, many many models…**



Compare models

o3-mini

GPT-4o

o1

Fast, flexible, intelligent reasoning model

Fast, intelligent, flexible GPT model

Reasoning ●●●●

Speed ⚡⚡⚡

Intelligence ●●●

Speed ⚡⚡⚡

Speed ⚡

Trivia: Can ChatGPT Answer Trivia Questions?

- Collected 50K trivia questions (multiple-choice questions – most 4 choices, some true/false)
- How accurate is technique based on word2vec (i.e., “a pretty good technique” prior to ChatGPT)?
  - ○ Can answer most of the questions perfectly
  - ○ Fairly good
  - ○ Very bad (even worse than randomly guessing)

[https://www.sliceofexperiments.com/p/chatgpt-vs-50000-trivia-questions]

Trivia: Can ChatGPT Answer Trivia Questions?

- Collected 50K trivia questions (multiple-choice questions – most 4 choices, some true/false)
- How accurate is technique based on word2vec (i.e., “a pretty good technique” prior to ChatGPT)?
  - ○ Can answer most of the questions perfectly
  - ○ Fairly good
  - ○ **Very bad (even worse than randomly guessing)**

[https://www.sliceofexperiments.com/p/chatgpt-vs-50000-trivia-questions]

Trivia: Can ChatGPT Answer Trivia Questions?

- How accurate is ChatGPT?
  - ○ 99.5%
  - ○ 82.9%
  - ○ 66.7%
  - ○ 35.5%
  - ○ Very bad (even worse than randomly guessing)

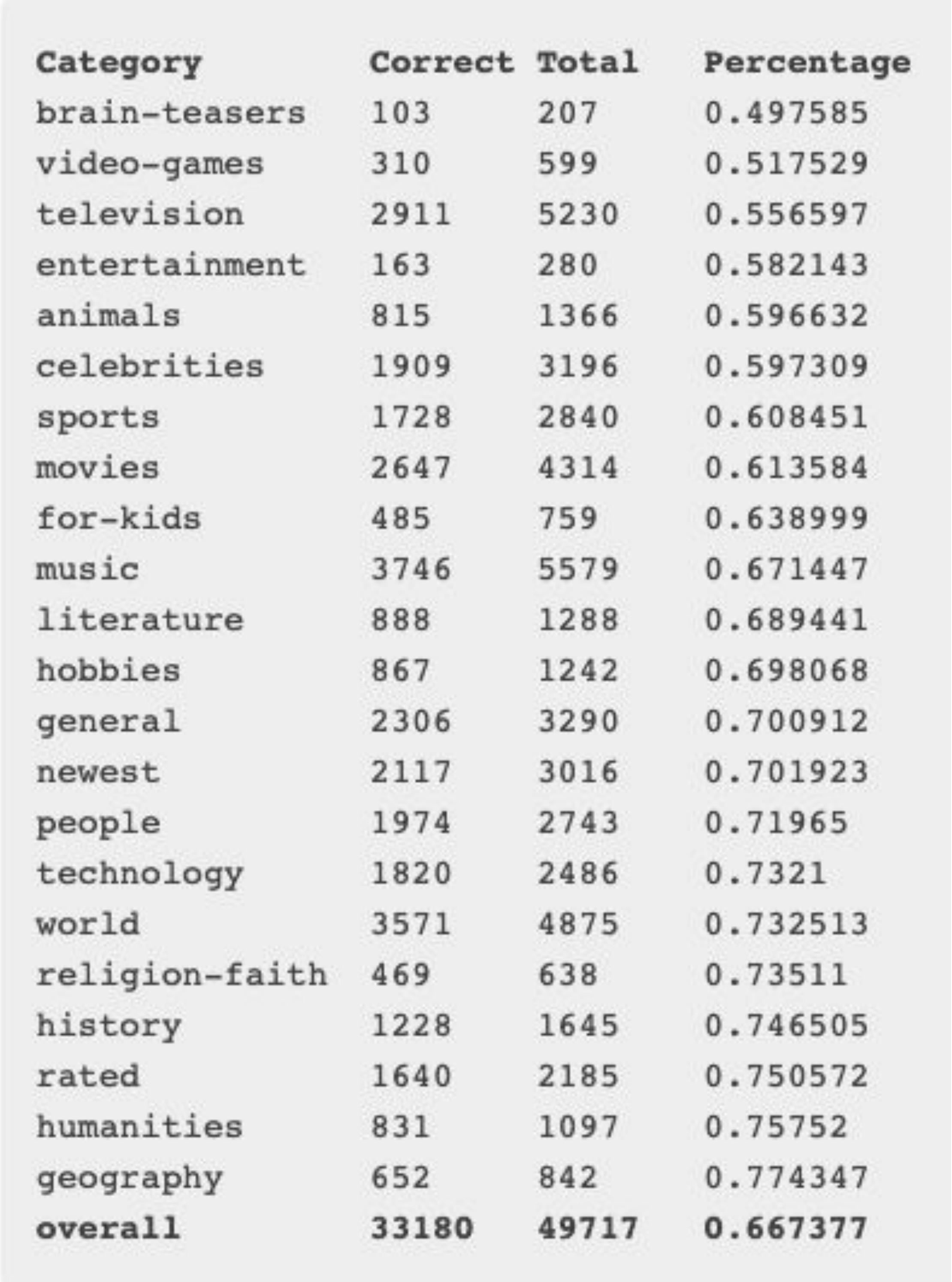
[https://www.sliceofexperiments.com/p/chatgpt-vs-50000-trivia-questions]

Trivia: Can ChatGPT Answer Trivia Questions?

- How accurate is ChatGPT?
  - 99.5%
  - 82.9%
  - **66.7%**
  - 35.5%
  - Very bad (even worse than randomly guessing)

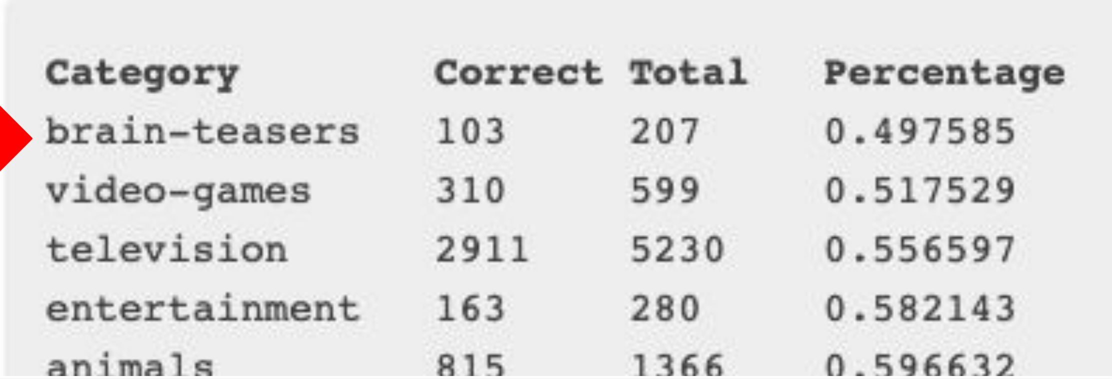
[https://www.sliceofexperiments.com/p/chatgpt-vs-50000-trivia-questions]

Trivia: Can ChatGPT Answer

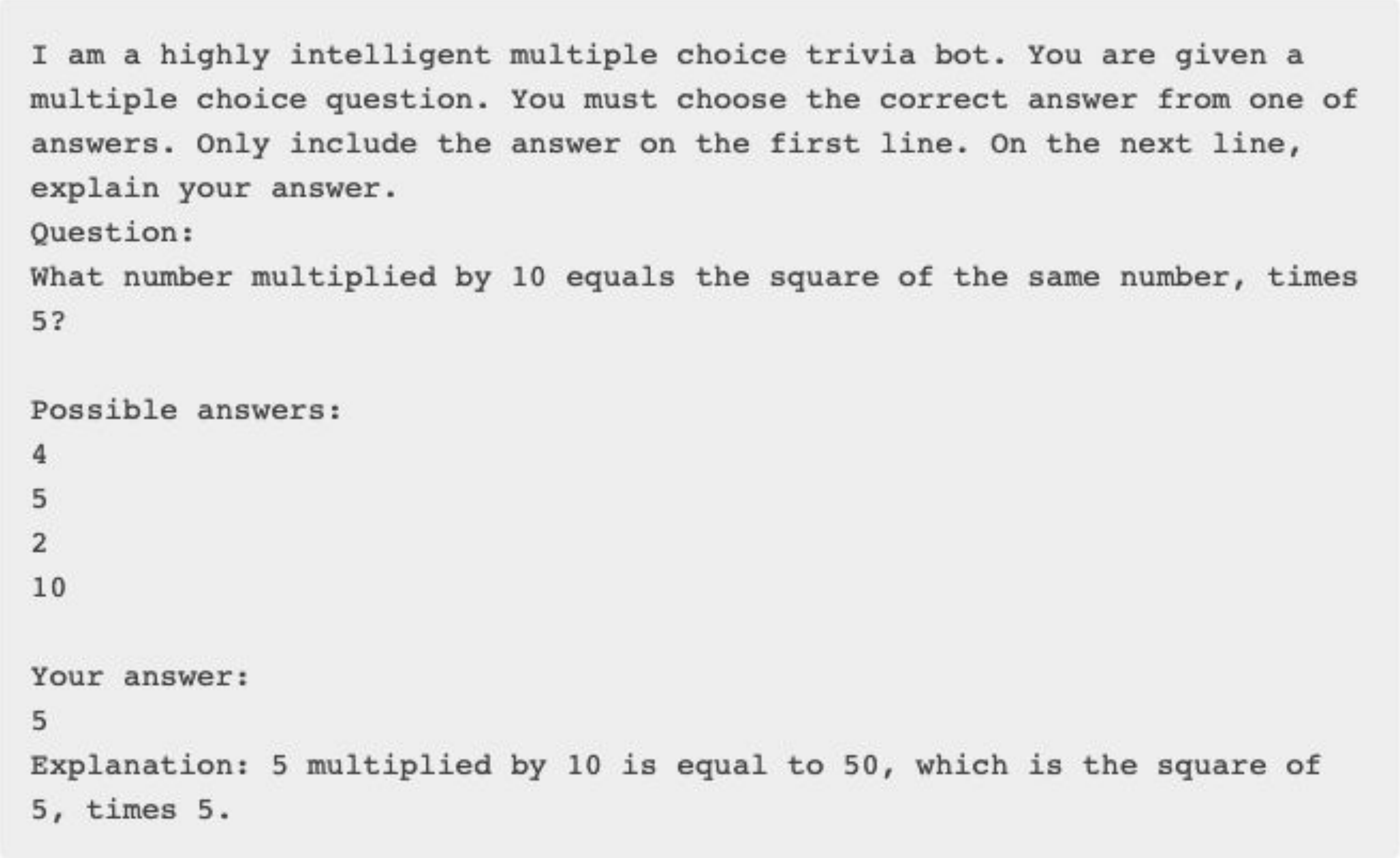


Category	Correct	Total	Percentage
brain-teasers	103	207	0.497585
video-games	310	599	0.517529
television	2911	5230	0.556597
entertainment	163	280	0.582143
animals	815	1366	0.596632
celebrities	1909	3196	0.597309
sports	1728	2840	0.608451
movies	2647	4314	0.613584
for-kids	485	759	0.638999
music	3746	5579	0.671447
literature	888	1288	0.689441
hobbies	867	1242	0.698068
general	2306	3290	0.700912
newest	2117	3016	0.701923
people	1974	2743	0.71965
technology	1820	2486	0.7321
world	3571	4875	0.732513
religion-faith	469	638	0.73511
history	1228	1645	0.746505
rated	1640	2185	0.750572
humanities	831	1097	0.75752
geography	652	842	0.774347
overall	**33180**	**49717**	**0.667377**

Trivia: Can ChatGPT Answe



Category	Correct	Total	Percentage
brain-teasers	103	207	0.497585
video-games	310	599	0.517529
television	2911	5230	0.556597
entertainment	163	280	0.582143
animals	815	1366	0.596632

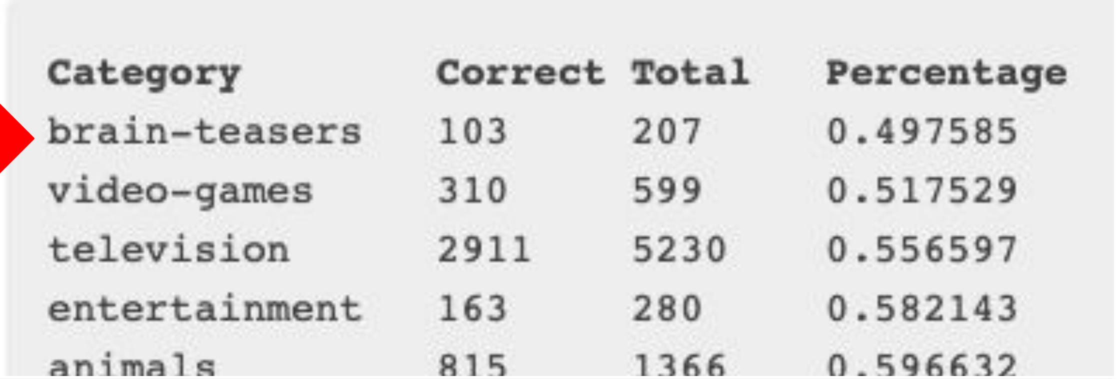


```
I am a highly intelligent multiple choice trivia bot. You are given a
multiple choice question. You must choose the correct answer from one of
answers. Only include the answer on the first line. On the next line,
explain your answer.
Question:
What number multiplied by 10 equals the square of the same number, times
5?

Possible answers:
4
5
2
10

Your answer:
5
Explanation: 5 multiplied by 10 is equal to 50, which is the square of
5, times 5.
```

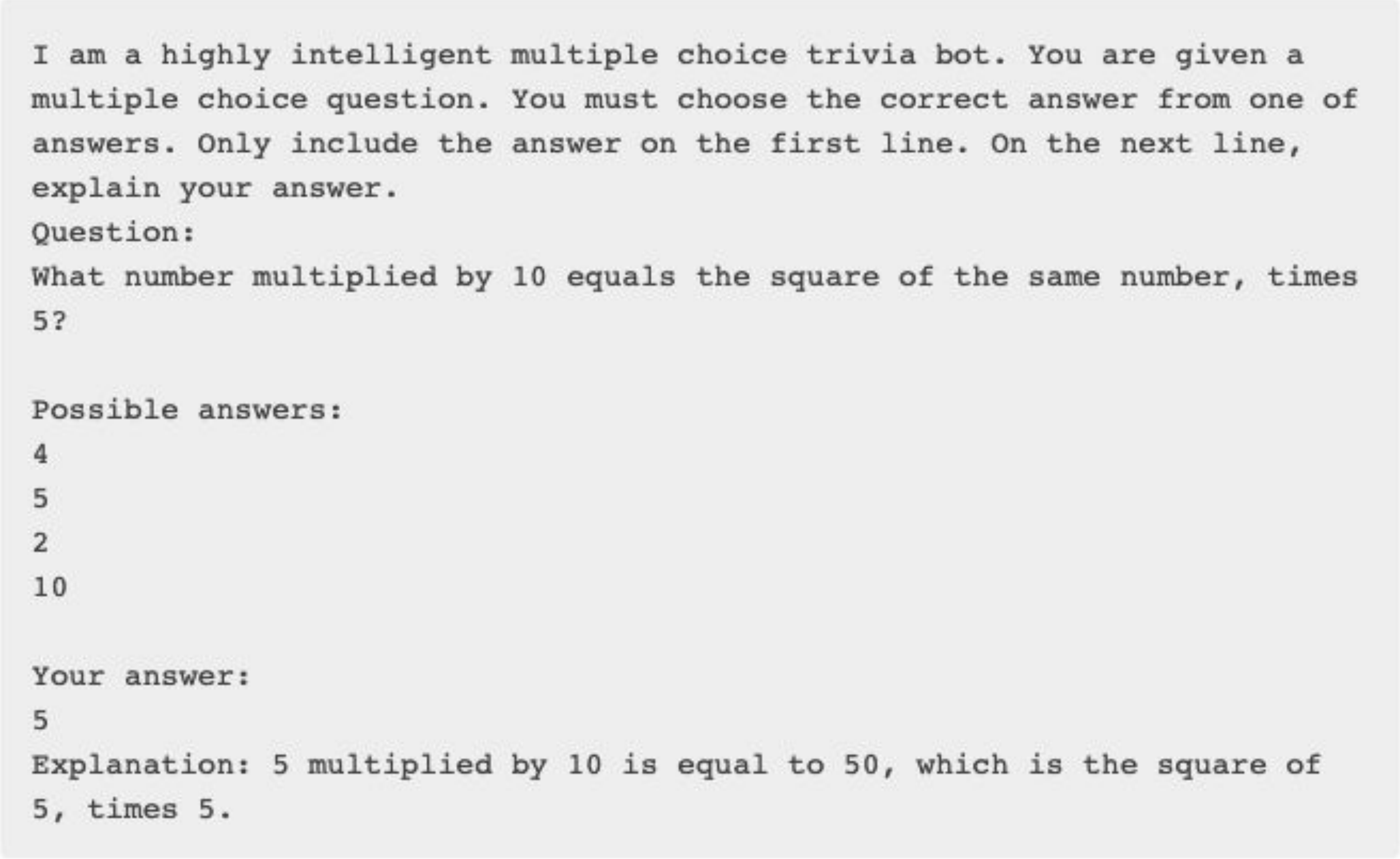
Trivia: Can ChatGPT Answe



Category	Correct	Total	Percentage
brain-teasers	103	207	0.497585
video-games	310	599	0.517529
television	2911	5230	0.556597
entertainment	163	280	0.582143
animals	815	1366	0.596632

ChatGPT cannot do arithmetic/math.

Can ChatGPT "repair programs"?



```
I am a highly intelligent multiple choice trivia bot. You are given a
multiple choice question. You must choose the correct answer from one of
answers. Only include the answer on the first line. On the next line,
explain your answer.
Question:
What number multiplied by 10 equals the square of the same number, times
5?

Possible answers:
4
5
2
10

Your answer:
5
Explanation: 5 multiplied by 10 is equal to 50, which is the square of
5, times 5.
```

Trivia: Can ChatGPT Find/Fix Program Bugs?

- Out of total 40 buggy programs, how many can ChatGPT fix? And how many can a standard technique (like GenProg) fix?
  - ○ ChatGPT/Standard = 10/35
  - ○ ChatGPT/Standard = 19/21
  - ○ ChatGPT/Standard = 28/12
  - ○ ChatGPT/Standard = 31/7

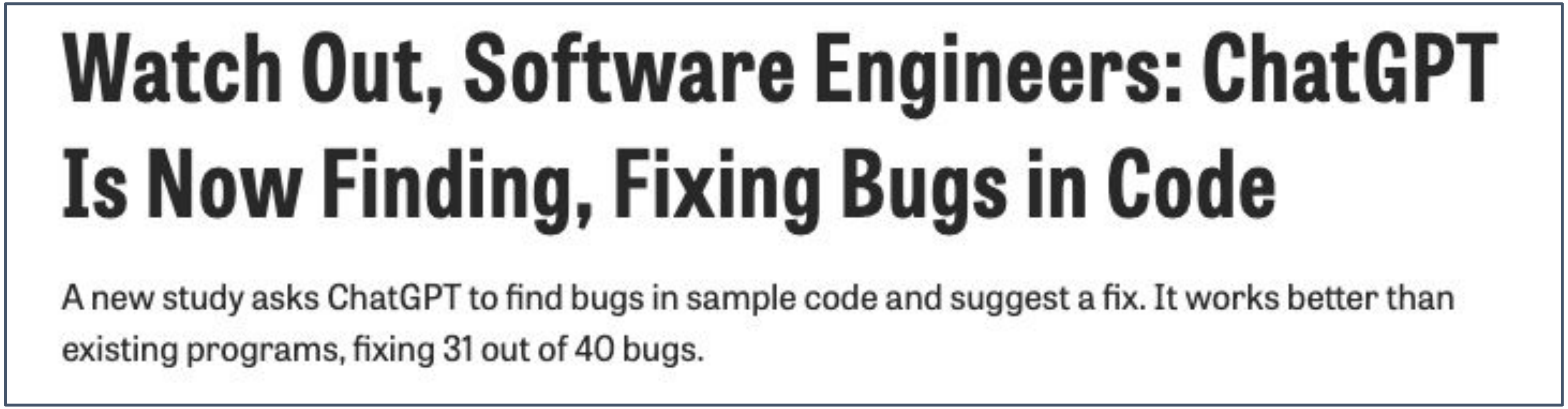
[https://www.pcmag.com/news/watch-out-software-engineers-chatgpt-is-now-finding-fixing-bugs-in-code]

Trivia: Can ChatGPT Find/Fix Program Bugs?

- Out of total 40 buggy programs, how many can ChatGPT fix? And how many can a standard technique (like GenProg) fix?
  - ○ ChatGPT/Standard = 10/35
  - ○ ChatGPT/Standard = 19/21
  - ○ ChatGPT/Standard = 28/12
  - ○ **ChatGPT/Standard = 31/7**

[https://www.pcmag.com/news/watch-out-software-engineers-chatgpt-is-now-finding-fixing-bugs-in-code]

Trivia: Can ChatGPT Find/Fix Program Bugs?

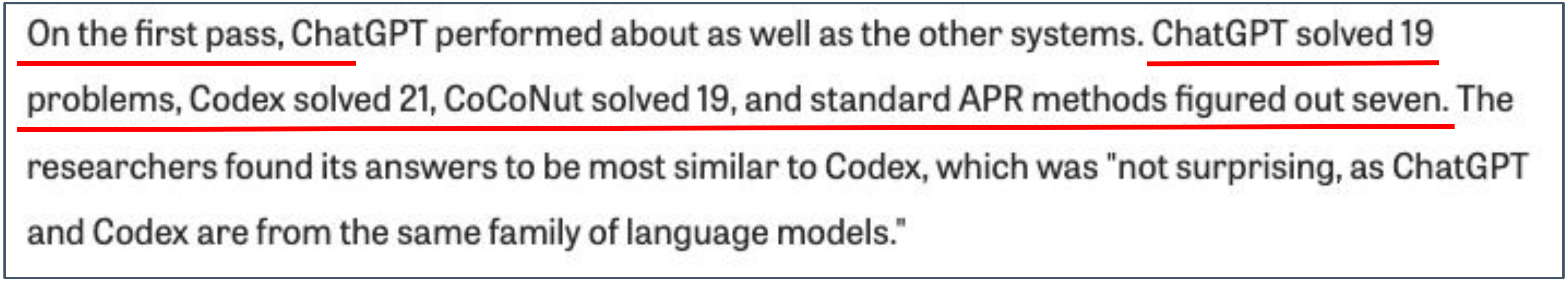


Watch Out, Software Engineers: ChatGPT Is Now Finding, Fixing Bugs in Code

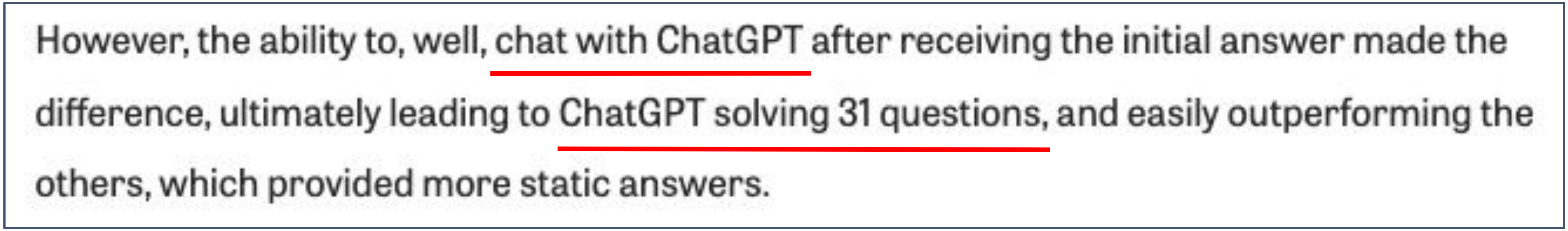
A new study asks ChatGPT to find bugs in sample code and suggest a fix. It works better than existing programs, fixing 31 out of 40 bugs.

[https://www.pcmag.com/news/watch-out-software-engineers-chatgpt-is-now-finding-fixing-bugs-in-code]

Can ChatGPT Find/Fix Program Bugs?

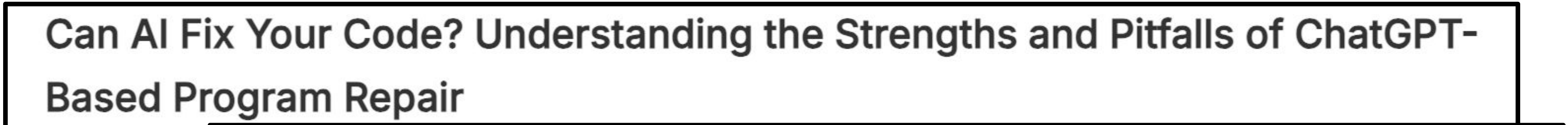


On the first pass, ChatGPT performed about as well as the other systems. ChatGPT solved 19 problems, Codex solved 21, CoCoNut solved 19, and standard APR methods figured out seven. The researchers found its answers to be most similar to Codex, which was "not surprising, as ChatGPT and Codex are from the same family of language models."

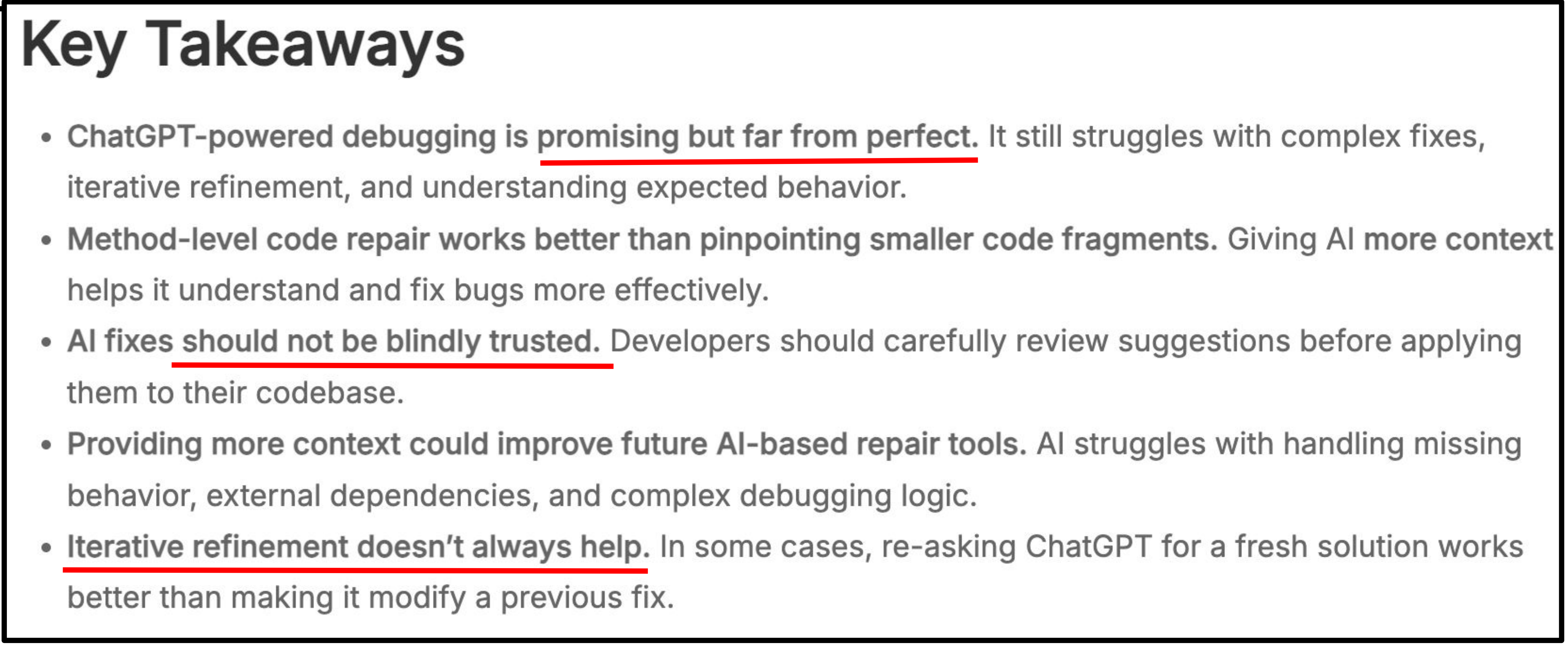


However, the ability to, well, chat with ChatGPT after receiving the initial answer made the difference, ultimately leading to ChatGPT solving 31 questions, and easily outperforming the others, which provided more static answers.

A More Recent Study… (March 2025)



Can AI Fix Your Code? Understanding the Strengths and Pitfalls of ChatGPT-Based Program Repair



Key Takeaways

- **ChatGPT-powered debugging is promising but far from perfect.** It still struggles with complex fixes, iterative refinement, and understanding expected behavior.
- **Method-level code repair works better than pinpointing smaller code fragments.** Giving AI **more context** helps it understand and fix bugs more effectively.
- **AI fixes should not be blindly trusted.** Developers should carefully review suggestions before applying them to their codebase.
- **Providing more context could improve future AI-based repair tools.** AI struggles with handling missing behavior, external dependencies, and complex debugging logic.
- **Iterative refinement doesn't always help.** In some cases, re-asking ChatGPT for a fresh solution works better than making it modify a previous fix.

[https://theministryofai.org/can-ai-fix-your-code-understanding-the-strengths-and-pitfalls-of-chatgpt-based-program-repair/]

Relationship with Mutation Testing

- This program repair approach is a **dual** of **mutation testing**
  - This suggests avenues for cross-fertilization and helps explain some of the successes and failures of program repair.
- Very (in)formally:
  - PR **Exists** M in Mut. **Forall** T in Tests. M(T)
  - MT **Forall** M in Mut. **Exists** T in Tests. Not M(T)

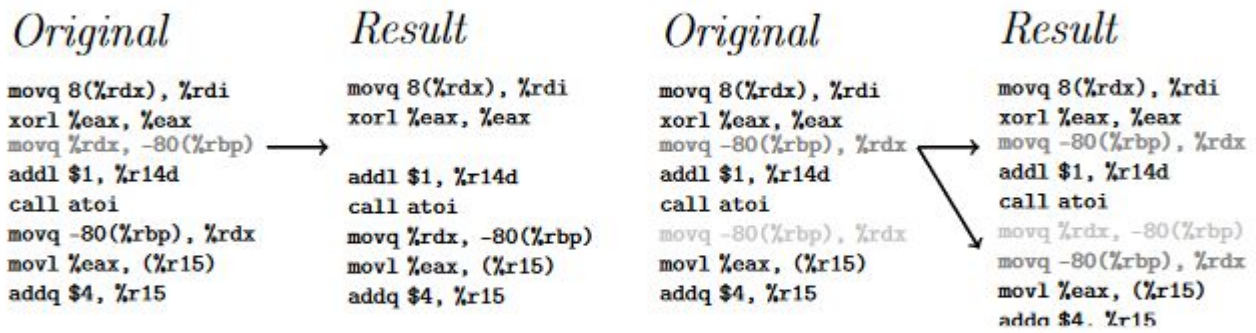
Idealized Formulation

- Ideally, mutation testing takes a program that **passes** its test suite and requires that **all** mutants *(based on human* ***mistakes*** *from the* ***entire*** *program that are not equivalent)* **fail** at least one test.

- By contrast, program repair takes a program that **fails** its test suite and requires that **one** mutant *(based on human* ***repairs*** *from the fault* ***localization*** *only)* **pass** all tests.

No Source Code Needed

- Can repair assembly or binary programs to support **multi-language projects**



```
movq 8(%rdx), %rdi
xorl %eax, %eax
movq %rdx, -80(%rbp)
addl $1, %r14d
call atoi
movq -80(%rbp), %rdx
movl %eax, (%r15)
addq $4, %r15
```

```
movq 8(%rdx), %rdi
xorl %eax, %eax

addl $1, %r14d
call atoi
movq %rdx, -80(%rbp)
movl %eax, (%r15)
addq $4, %r15
```

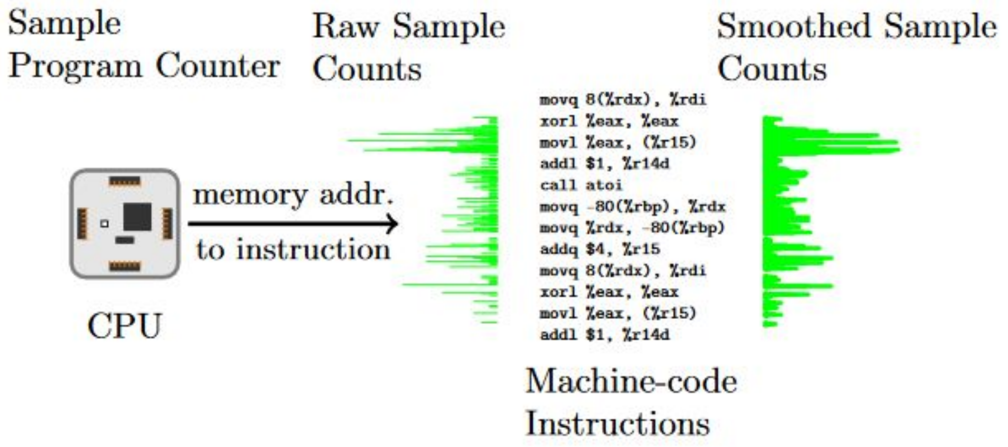
(a) Delete

```
movq 8(%rdx), %rdi
xorl %eax, %eax
movq -80(%rbp), %rdx
addl $1, %r14d
call atoi
movq -80(%rbp), %rdx
movl %eax, (%r15)
addq $4, %r15
```

```
movq 8(%rdx), %rdi
xorl %eax, %eax
movq -80(%rbp), %rdx
addl $1, %r14d
call atoi
movq %rdx, -80(%rbp)
movq -80(%rbp), %rdx
movl %eax, (%r15)
addq $4, %r15
```

(b) Insert

- Use **sampling-based profiling** for fault localization

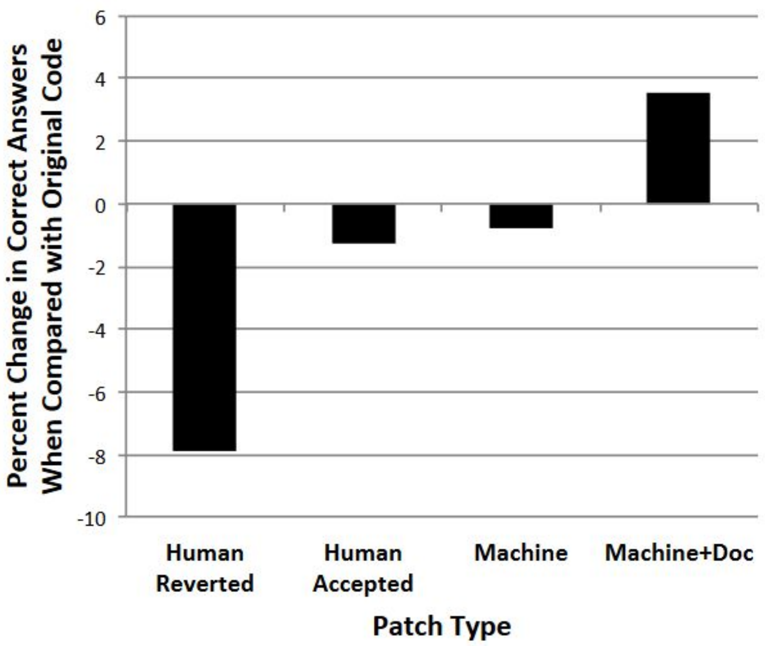


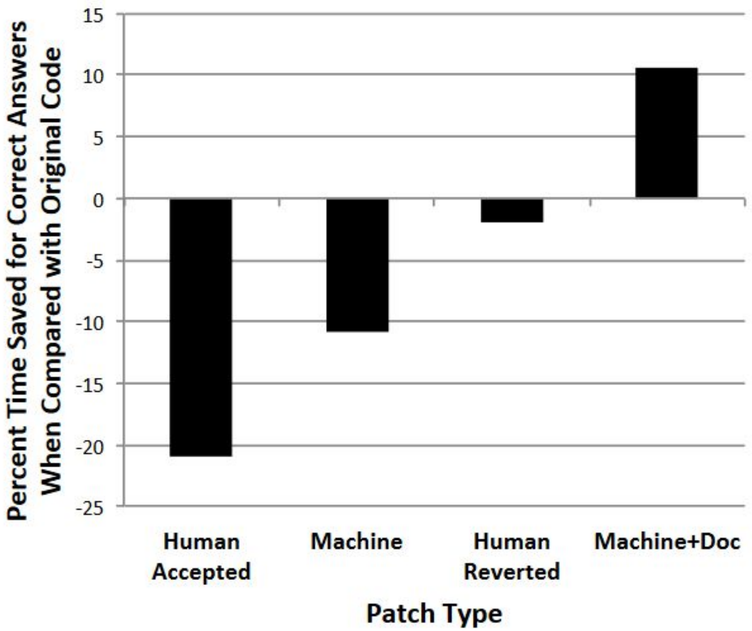
```
movq 8(%rdx), %rdi
xorl %eax, %eax
movl %eax, (%r15)
addl $1, %r14d
call atoi
movq -80(%rbp), %rdx
movq %rdx, -80(%rbp)
addq $4, %r15
movq 8(%rdx), %rdi
xorl %eax, %eax
movl %eax, (%r15)
addl $1, %r14d
```

[Schulte et al. *Automated Program Repair of Binary and Assembly Programs for Cooperating Embedded Devices.*]

Can Humans Use These Patches? (Usability)

- Synthesize “What” comments for generated patches (**design for maintainability**)
  - **Test input generation** constraints → English
  - Human study (N=150): “With docs → Yes!”





[Fry et al. *A Human Study of Patch Maintainability.*]

Human-Machine Partnerships

- What if your partner in **pair programming** were a machine that suggested patches?
  - Machine is driver, you are navigator/observer
  - In response to your feedback and characterization of program state, it suggests new patches
- You note "checkpoints" where at point *X*, test *Y* is running correctly (or variable *Z* is wrong)
- Human study of first-year grads (N=25):
  - Reduces debugging on 14/15 scenarios, compared to one person working alone (~60% reduction over all 15)

[Xinrui Guo. *SmartDebug: An Interactive Debug Assistant for Java.*]

Repair Concurrency Bugs?

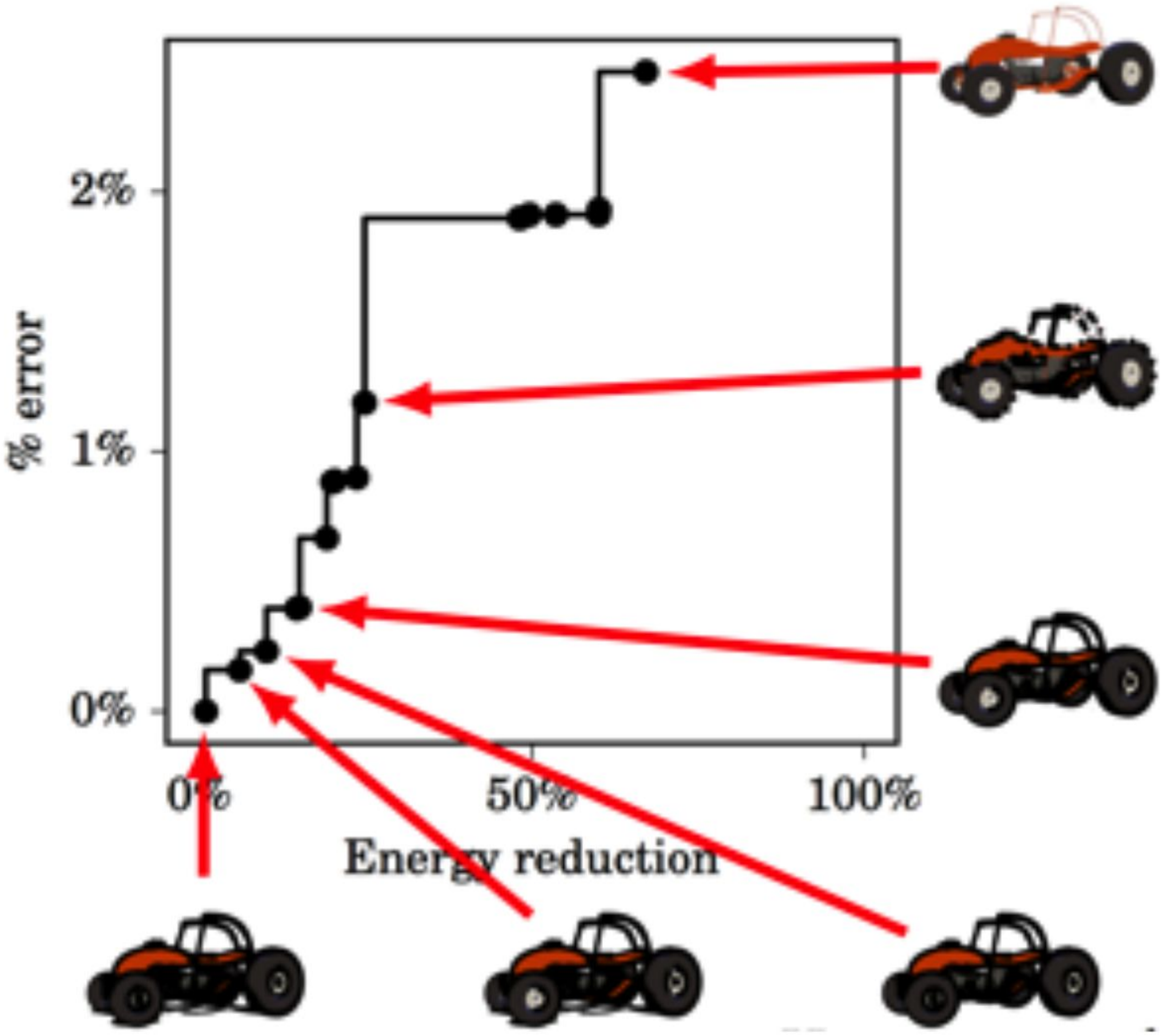
- So far we have required **deterministic** tests (no concurrency in program)
- We can use a **dynamic analysis** like CHESS or Eraser to detect **concurrency** bugs
  - Look for two threads accessing *X*, one is a write
- Use special repair templates (e.g., always add paired lock()/unlock() calls)
- Fixes 6/8 historical single-variable atomicity violations in Apache, MySQL, Mozilla, etc.
  - Devs fixed 6/8 in 11 days each, on average
  - Union of both fixes all 8/8

[Jin et al. *Automated Atomicity-Violation Fixing.*]

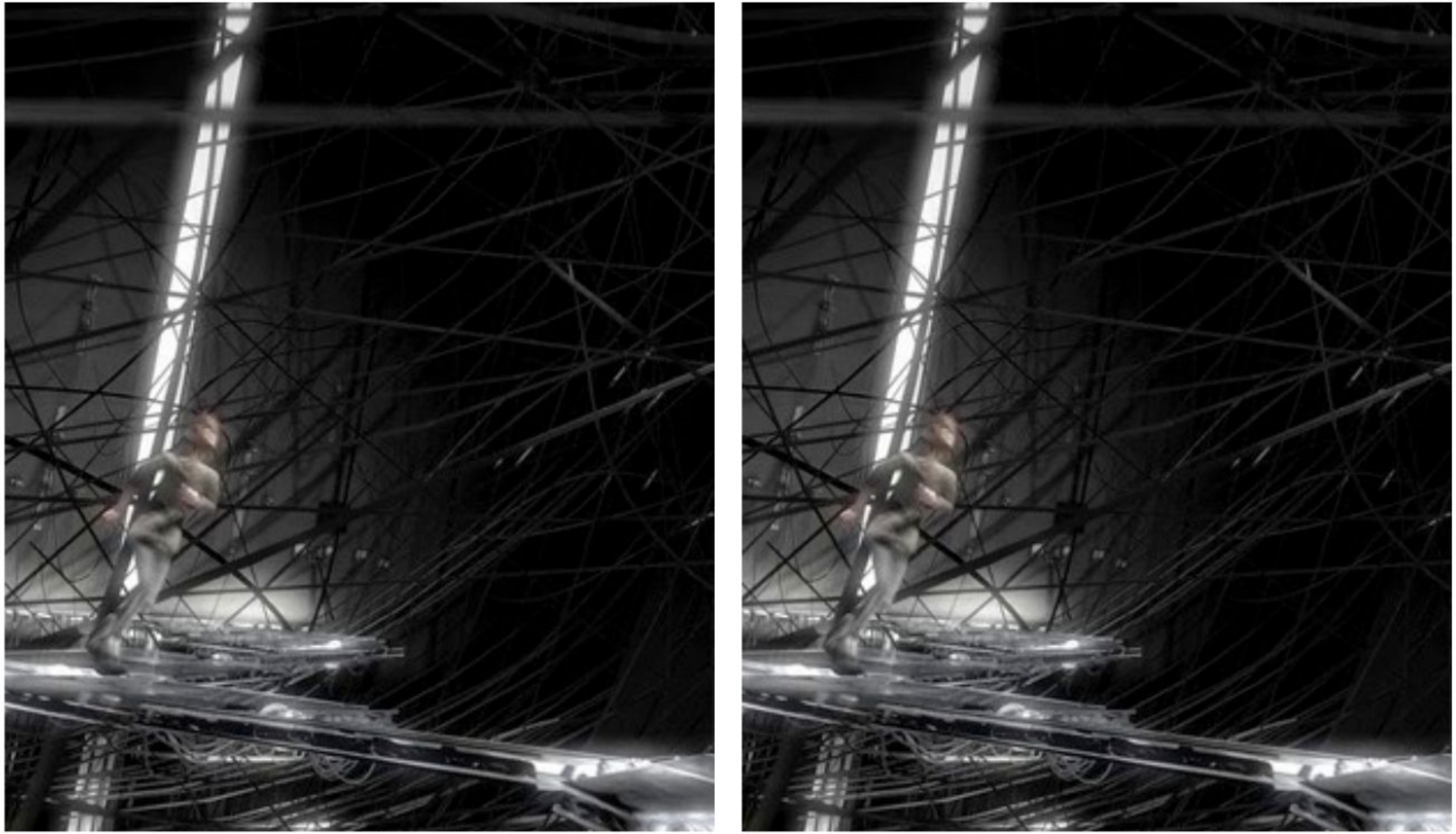
Repair Quality (Non-Functional) Defects?

- What if the bug is that your program is too slow (aka. performance bug) *or* too big *or* uses too much energy?
- We can also improve and trade-off **verifiable quality properties** (requirements solicitation)
  - cf. MP3 or JPG *lossy* compression: space vs. quality
- Candidates must pass all functional tests
- But we also measure quality properties of all passing candidates
- Present a Pareto frontier to help user explore alternative solutions to requirement conflicts

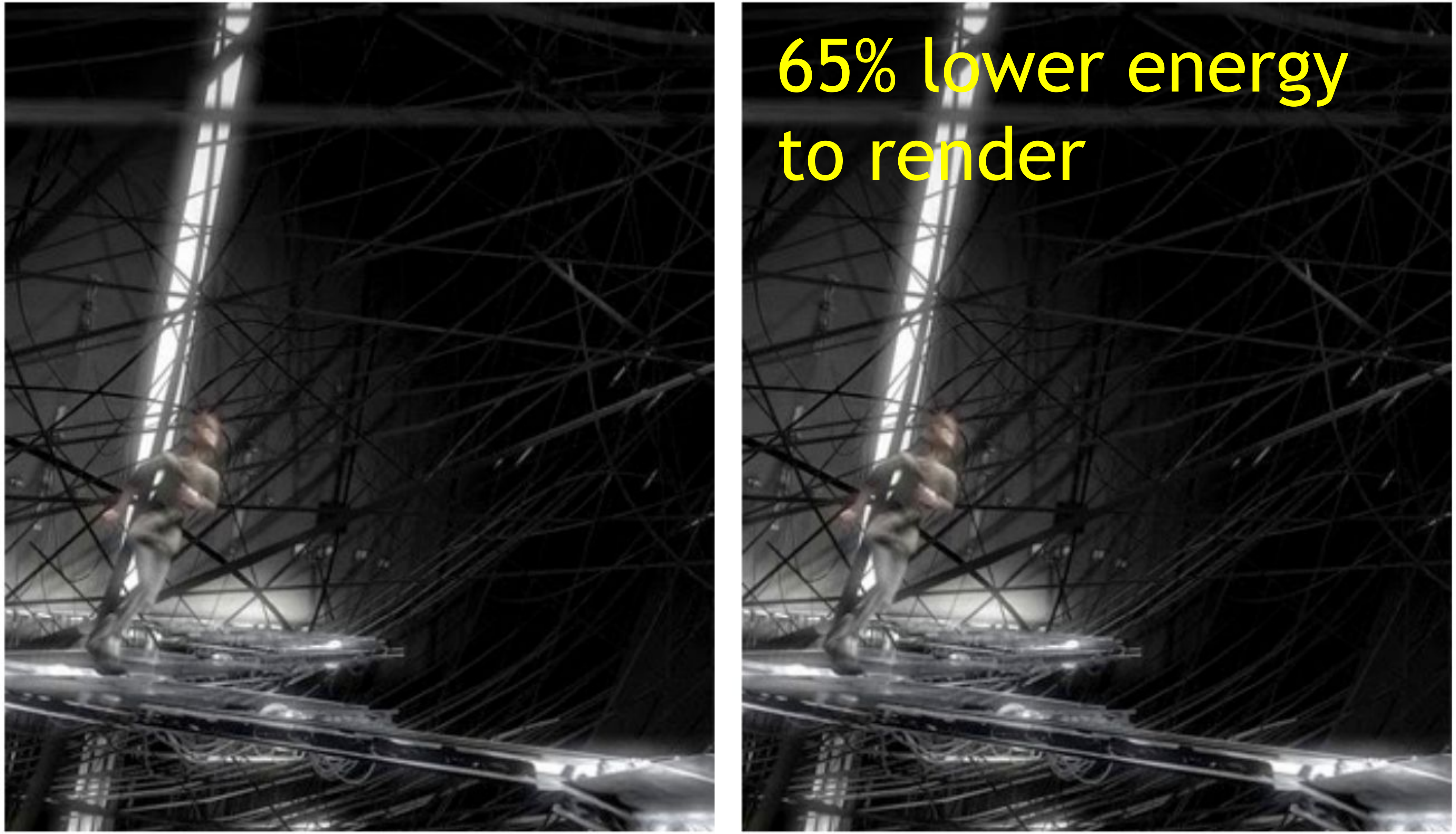
Automatically Exploring Tradeoffs In Conflicting Requirements



Can you spot the difference?



Can you spot the difference?



65% lower energy to render

[Dorn et al. *Automatically exploring tradeoffs between software output fidelity and energy costs.*]

“Wishes Come True, Not Free”

- Automated program repair, the whiny child:
  - “You only said I had to *get in* to the bathtub, you didn't say I had to wash.”
- The **specification** (tests) must encode **requirements** (cf. conflicts)
- GenProg's first webserver defect repair
  - 5 regression tests (GET index.html, etc.)
  - 1 bug (POST ⟶ remote security exploit)
  - GenProg's fix: remove POST functionality
  - (Adding a 6th test yields a high-quality repair.)
  - **Humans write high-quality patch (before) -> high-quality test (now)**

Requirements and Testing

- MIT Lincoln Labs evaluation of GenProg: sort
  - Tests: “the output of sort is in sorted order”
  - GenProg's fix: “always output the empty list”
  - (More tests yield a higher-quality repair. cf. design-by-contract pre- and post-conditions)
- Existing human-written tests suites implicitly assume the developers are reasonable humans
  - Unless you are outsourcing, you rarely test against “creative” for “adversarial” solutions or bugs
  - cf. “we're already good at this” denials, terminology conflicts

Measuring Quality via Tests

- Another GenProg example:
  - Tests: “compare yours.txt to trusted.txt”
  - GenProg's fix: “delete trusted.txt, output nothing”
- Canonical **perverse incentives** situation
  - Automated program repair optimizes the **metric**
  - “What you said” not “What you meant”
- Sleep forever to avoid CPU-usage penalties
- Always segfault to avoid bad output checks

[Weimer. *Advances in Automated Program Repair and a Call to Arms.*]

The Future

- Despite quality and trust concerns, some form of this is coming in the future (10-20 years?)
  - Already-demonstrated **productivity** gains
- What if “solve this one-line bug” became an atomic action in your lexicon?
  - The same way “complete this method call” or “sort” or “rename this variable” is today

Questions

- HW6b and Exam 2