Testing Oracles

Testing oracle- definition

"… the oracle can be a program specification, a table of examples, or simply the programmer's knowledge of how the program should operate…"

Howden

-a mechanism to evaluate the actual results of a test case as pass or no pass.

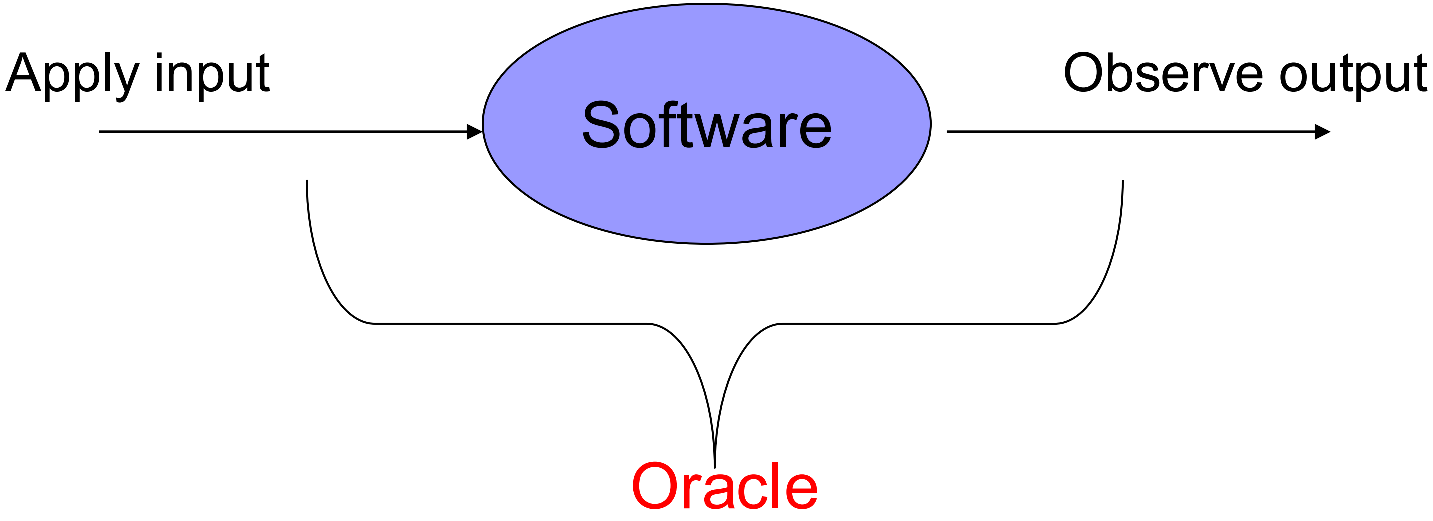
Two essential components:

•<u>result generator</u> (to produce the expected result for an input)

•<u>comparator</u> to check the actual results

Oracle = generation and comparison mechanism

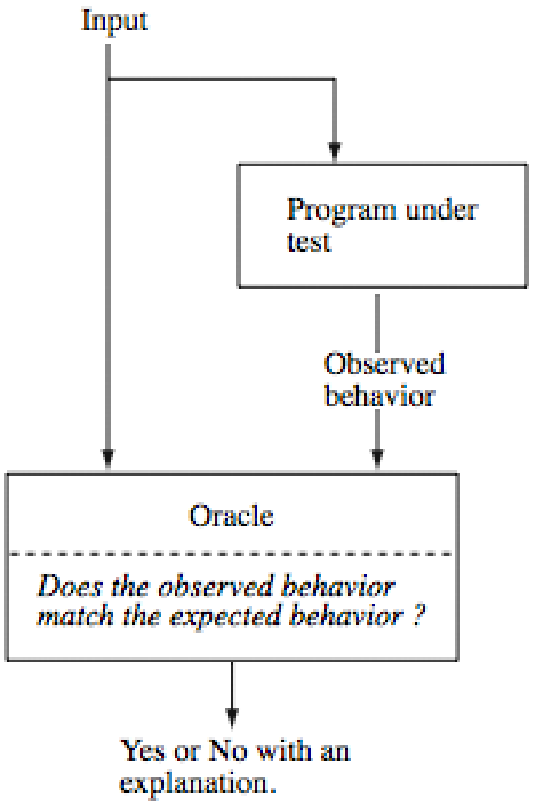
Testing



Validate the observed output against the expected output

Is the observed output the same as the expected output?

Oracle: Example (1)


Input
Program under test
Observed behavior
Oracle
Does the observed behavior match the expected behavior ?
Yes or No with an explanation.

How to verify the output of a matrix multiplication?

How to verify the output of a matrix inversion program?

How to verify the output of a sorting procedure?

Oracle: Example (2)

A tester assuming the role of an oracle and thus serving as human oracle.

How to verify the output of a matrix multiplication?

Hand calculation: the tester might input two matrices and check if the output of the program matches the results of hand calculation.

Oracles can also be programs. For example, one might use a matrix multiplication to check if a matrix inversion program has produced the correct result: $A \times A^{-1} = I$

How to verify the output of a sorting algorithm?

Oracle: Example (3)

Calculations of variance

$$s^2 = \frac{\sum_{i=1}^{n} (x_i - \overline{x})^2}{n-1}$$

Calculator formula (single pass through data)

$$\tilde{x} = x(1+\delta x), \tilde{y} = y(1+\delta y)$$

$$s^2 = \frac{\sum_{i=1}^{n} x_i^2 - \frac{1}{n}(\sum_{i=1}^{n} x_i)^2}{n-1}$$

Three data: 90,000,001 90,000,002 90,000,003; correct answer =1

Sum of squares 24,300,001,080,000,014

Second formula: the result is 4/(3-1)=2, incorrect

Catastrophic cancellation subtraction is ill-conditioned

Oracle: Example (3a)

Catastrophic cancellation

subtraction is ill-conditioned

$$\tilde{x} = x(1+\delta x), \tilde{y} = y(1+\delta y)$$

$$\tilde{x} - \tilde{y} = (x-y)\left(1 + \frac{x\delta x - y\delta y}{x-y}\right)$$

Testing oracle- further considerations

Oracles could be manual, automated, or partially automated

Perfect oracle – behaviorally equivalent to the implementation
(defect –free version of software under testing?)

Development of perfect oracle at least as difficult as developing original software

Main features (development requirements)

•Fidelity
•Generality (e.g., solved examples oracles)
•Cost

Testing oracle: Main categories

Judging oracles

Prespectification

- Solved example oracles
- Simulation oracles
- Approximation oracles

Gold standard oracles

- Trusted system oracle
- Parallel test oracle
- Voting oracle

Organic

- Smoke test
- Reversing oracles

Metamorphic testing

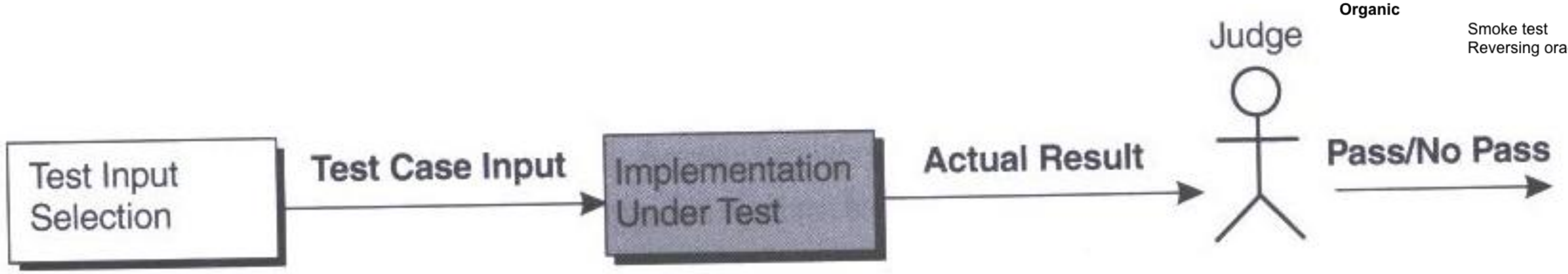
Judging oracles

Prespectification
- Solved example oracles
- Simulation oracles
- Approximation oracles

Gold standard oracles
- Trusted system oracle
- Parallel test oracle
- Voting oracle

Organic
- Smoke test
- Reversing oracles

Judging oracles



Making pass/no pass decision

Based on analysis, subjective evaluation, or both

Subjective evaluation – test cases improvised and results assessed on the fly (ad hoc beta testing). Judgment limited by human abilities; slow, error prone

Post-test analysis - test cases do not specify or provide only a general indication of expected results. The results are recorded for each test case and checked

Expert user - in case of qualitative evaluation (say, high resolution graphics, audio)

Judging oracles

Prespectification
- Solved example oracles
- Simulation oracles
- Approximation oracles

Gold standard oracles
- Trusted system oracle
- Parallel test oracle
- Voting oracle

Organic
- Smoke test
- Reversing oracles

Solved example oracles

Expected results are developed by hand

Each test case designed and output is prepared *manually*

The oracle useful at any scope but in general limited to evaluation of simple processing rules with several output variables

Used since the dawn of software development

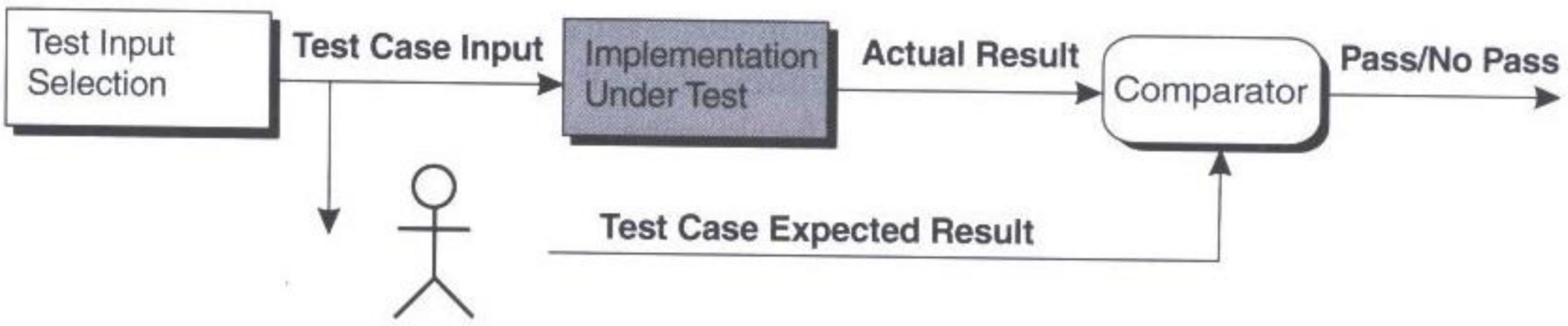
Judging oracles

Prespectification
- Solved example oracles
- Simulation oracles
- Approximation oracles

Gold standard oracles
- Trusted system oracle
- Parallel test oracle
- Voting oracle

Organic
- Smoke test
- Reversing oracles

Solved example oracles



Judging oracles

Prespecification
Solved example oracles
Simulation oracles
Approximation oracles

Gold standard oracles
Trusted system oracle
Parallel test oracle
Voting oracle

Organic
Smoke test
Reversing oracles

Simulation oracles

Expected results are produced using a reduced, simplified or prototype implementation of the system

A simulator should be designed so that its development and verification are easier than the original system (say, computing and search → use a spreadsheet)

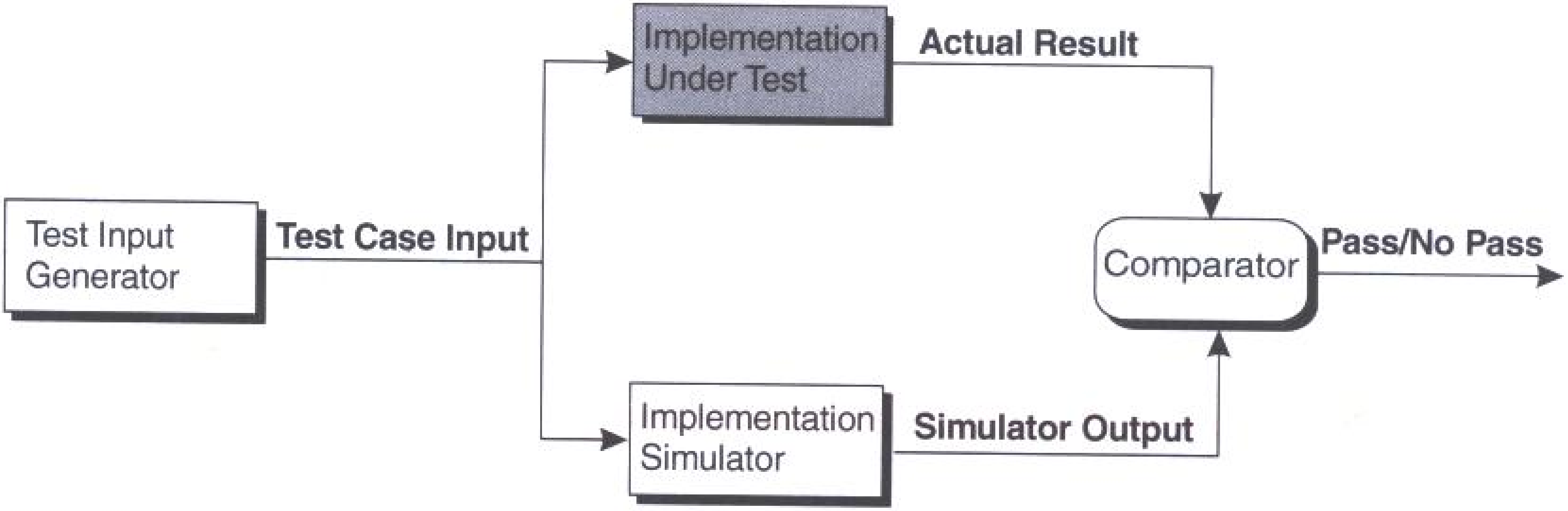
Simulation oracles

Judging oracles

Prespectification
- Solved example oracles
- Simulation oracles
- Approximation oracles

Gold standard oracles
- Trusted system oracle
- Parallel test oracle
- Voting oracle

Organic
- Smoke test
- Reversing oracles



Judging oracles

Prespectification
Solved example oracles
Simulation oracles
Approximation oracles

Gold standard oracles
Trusted system oracle
Parallel test oracle
Voting oracle

Organic
Smoke test
Reversing oracles

Approximation oracle

Systems built to solve unsolved problems (with no analytical solutions),
say weather/economic prediction, geological exploration, mechanical design

No simple techniques for checking the output produced by the system under testing

Some **strategies**
•**Reduce the input/output space**. Tests that are simple for which the answer can be checked by hand or simulation
In some systems, a **subject expert** may be able to recognize an output as wrong or Inconsistent
Constraints on some outputs ("x should never be greater than y", etc.…)
Determine the **acceptable accuracy** of the results; interpolate

Approximation oracle

Judging oracles

Prespectification

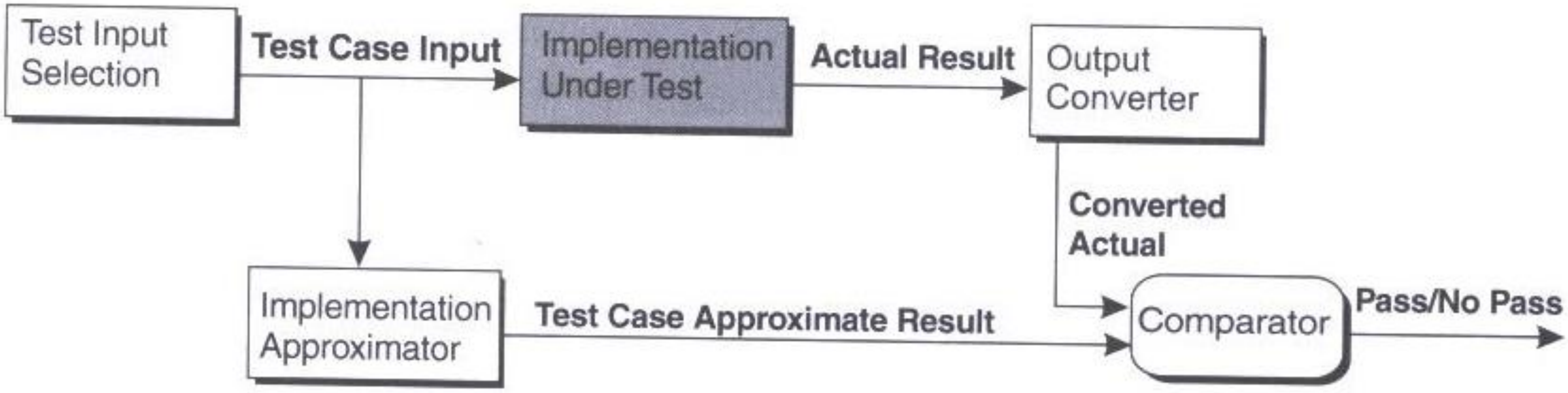
- Solved example oracles
- Simulation oracles
- Approximation oracles

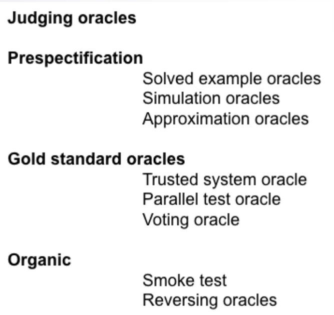
Gold standard oracles

- Trusted system oracle
- Parallel test oracle
- Voting oracle

Organic

- Smoke test
- Reversing oracles





Judging oracles

Prespecification
Solved example oracles
Simulation oracles
Approximation oracles

Gold standard oracles
Trusted system oracle
Parallel test oracle
Voting oracle

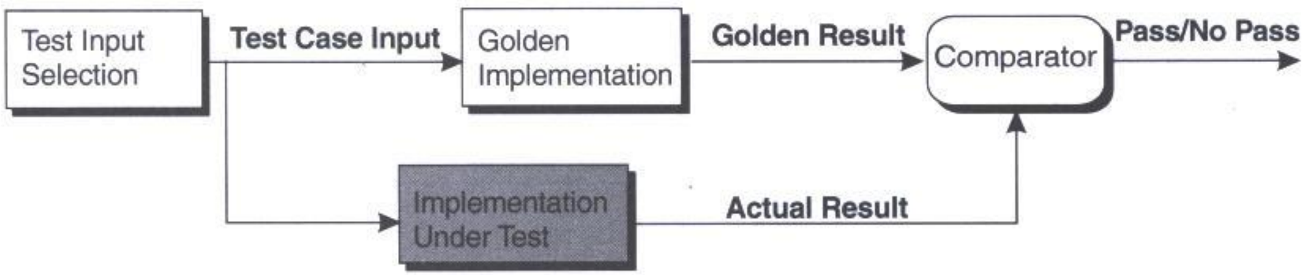
Organic
Smoke test
Reversing oracles

Gold standard oracles

Gold standard oracles use *one or more versions* of an existing application system to generate expected results

Several alternatives:

- **Trusted system oracle**
- **Parallel test oracle**
- **Voting oracle**



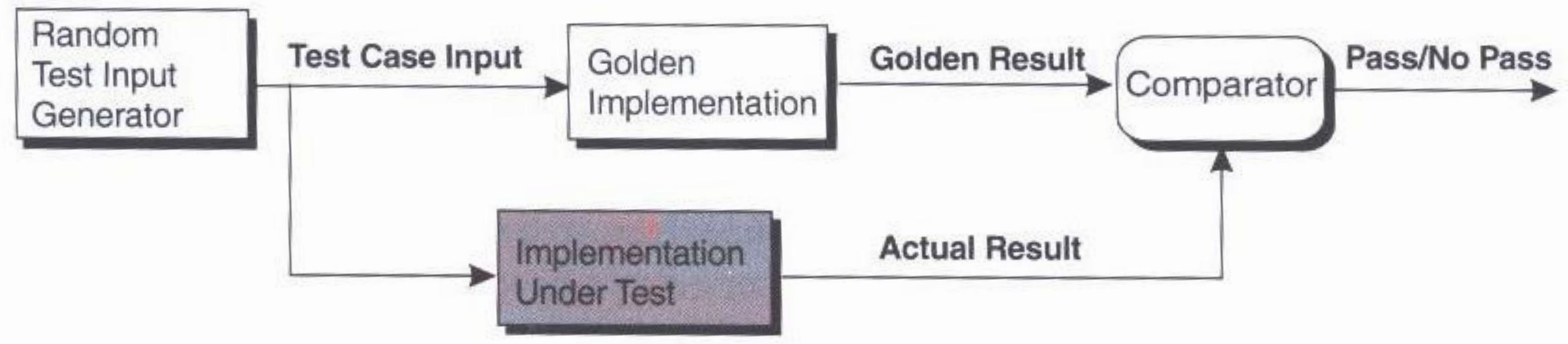
Gold standard oracles: trusted system oracle

Judging oracles

Prespectification
- Solved example oracles
- Simulation oracles
- Approximation oracles

Gold standard oracles
- Trusted system oracle
- Parallel test oracle
- Voting oracle

Organic
- Smoke test
- Reversing oracles



Existing system treated as trusted (routinely used with high confidence) has to compute many (if not all) of the same functions as the system under testing

Trusted system often present when dealing with legacy systems (to be replaced or ported); may need wrappers for them

Large quantity of test cases

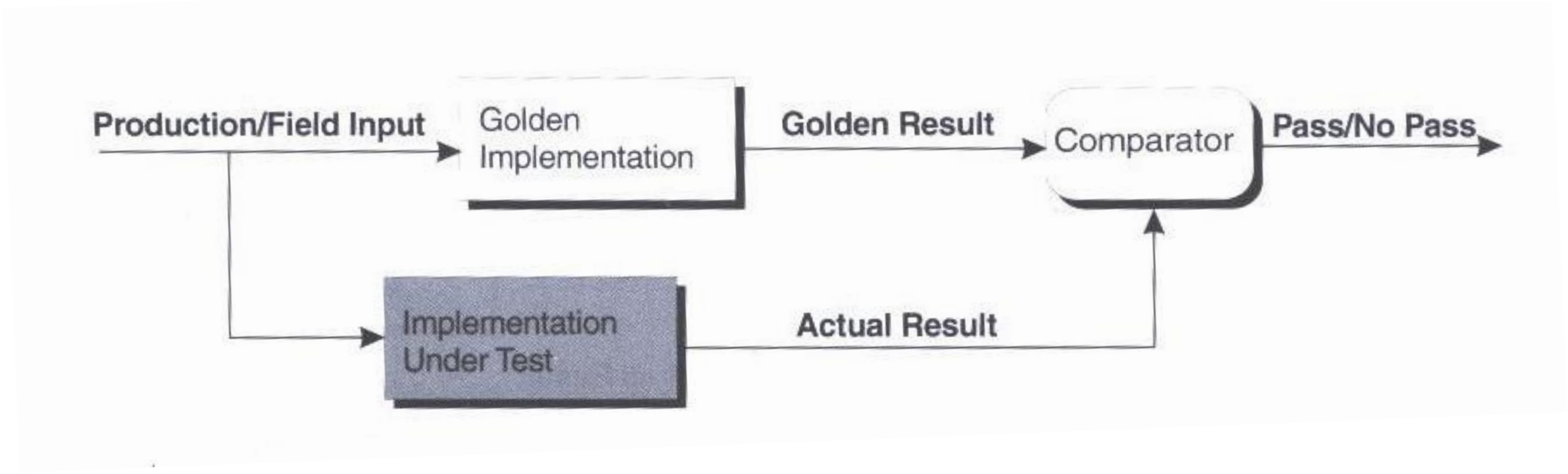
Gold standard oracles: parallel system oracle

Judging oracles

Prespectification
- Solved example oracles
- Simulation oracles
- Approximation oracles

Gold standard oracles
- Trusted system oracle
- Parallel test oracle
- Voting oracle

Organic
- Smoke test
- Reversing oracles



The use of the same input streams of data

Run on independent platforms; if not available, run each system in sequence so that there is no interaction

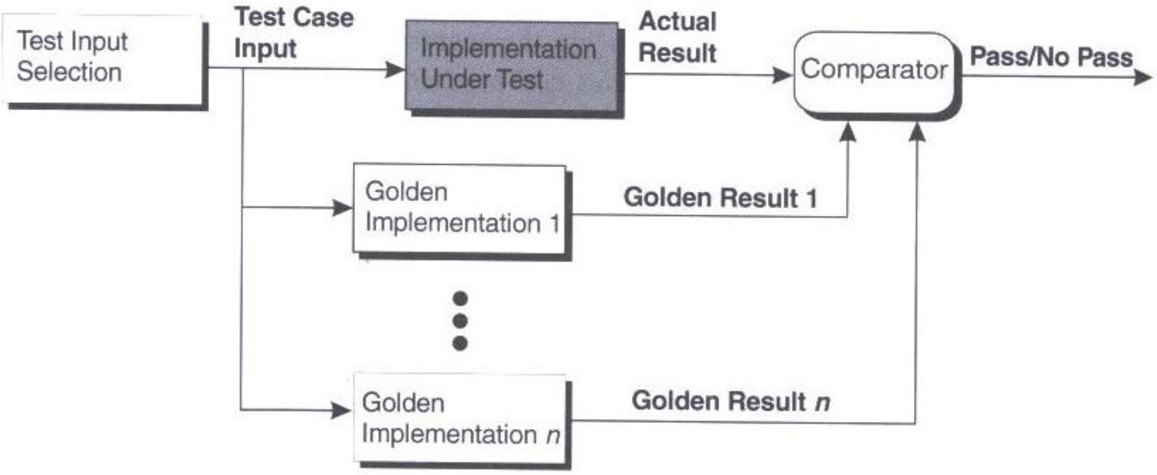
Gold standard oracles: voting oracle

Judging oracles

Prespectification
Solved example oracles
Simulation oracles
Approximation oracles

Gold standard oracles
Trusted system oracle
Parallel test oracle
Voting oracle

Organic
Smoke test
Reversing oracles



The use of two or more golden standards

Limited feasibility; need older versions of the trusted system as voters

May need data converters for input and output (the converters should be verified)

Smoke tests

Judging oracles

Prespecification
Solved example oracles
Simulation oracles
Approximation oracles

Gold standard oracles
Trusted system oracle
Parallel test oracle
Voting oracle

Organic
Smoke test
Reversing oracles

Usage of the basic operability checks of the runtime environment

The test does not require any expected results. Any test case that runs without abnormal termination is considered to have passed

Useful at any scope to establish minimal operability

Limited value

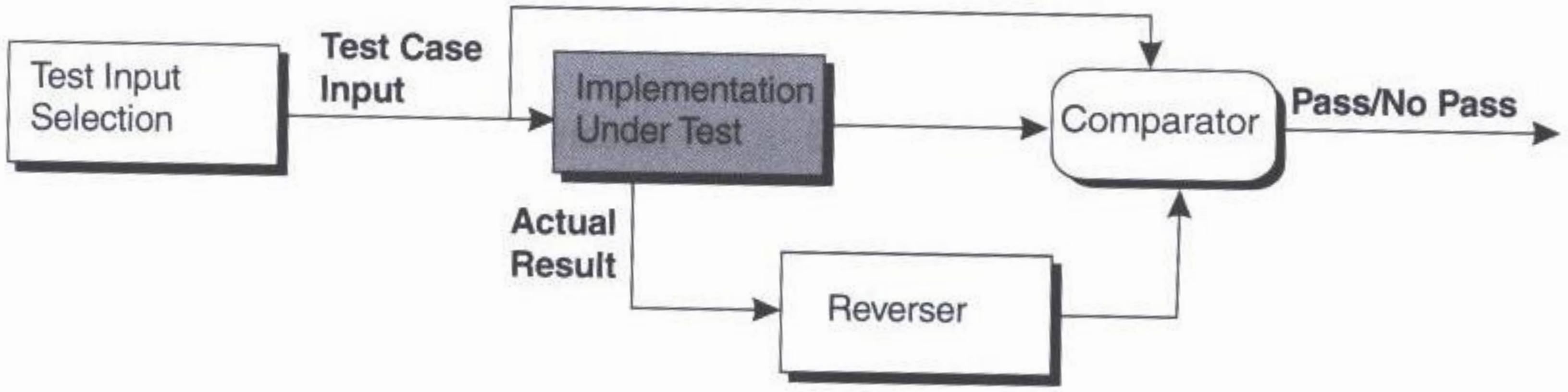
Reversing oracles

Some computations could be reversible [quite unusual]

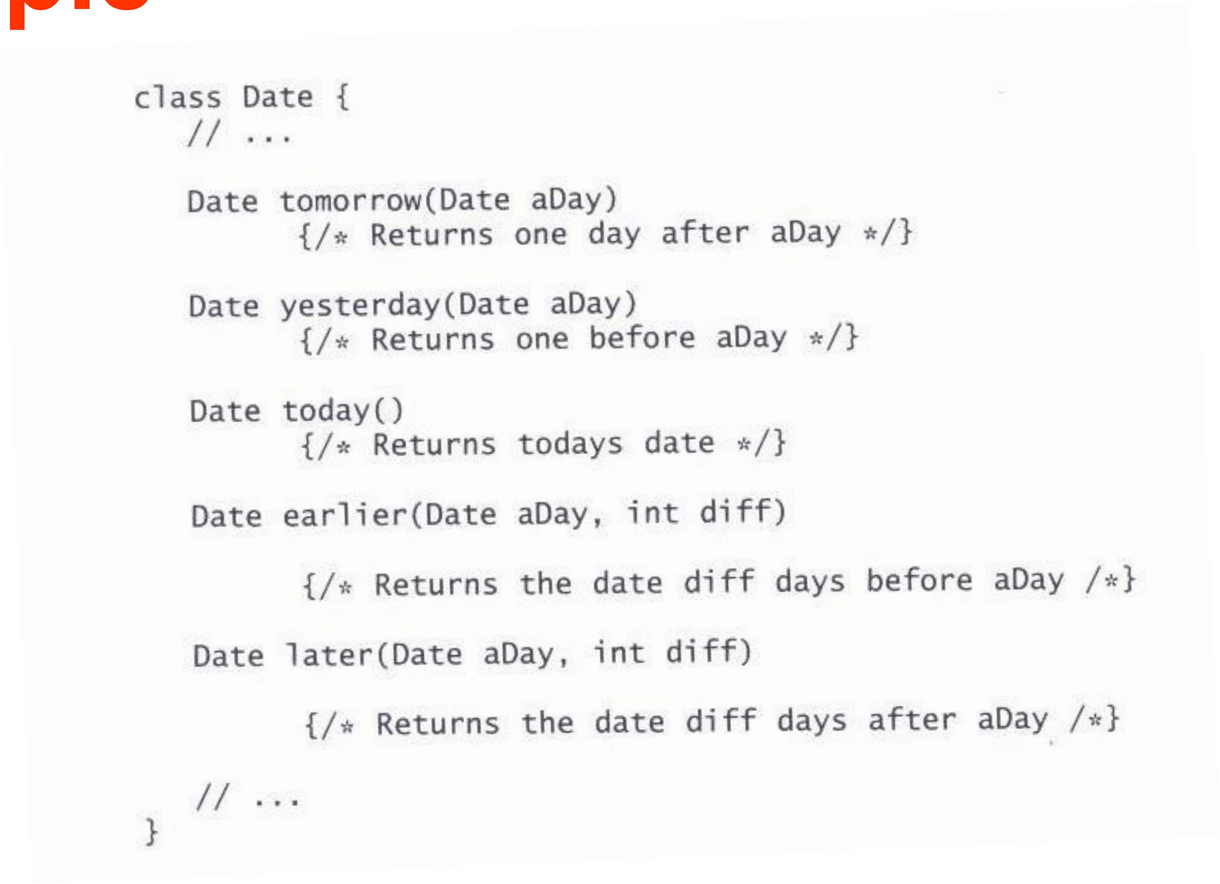
square root function vs. sorted list

If available could be exploited as oracles

No expected results need to be provided



Reversing oracles: example

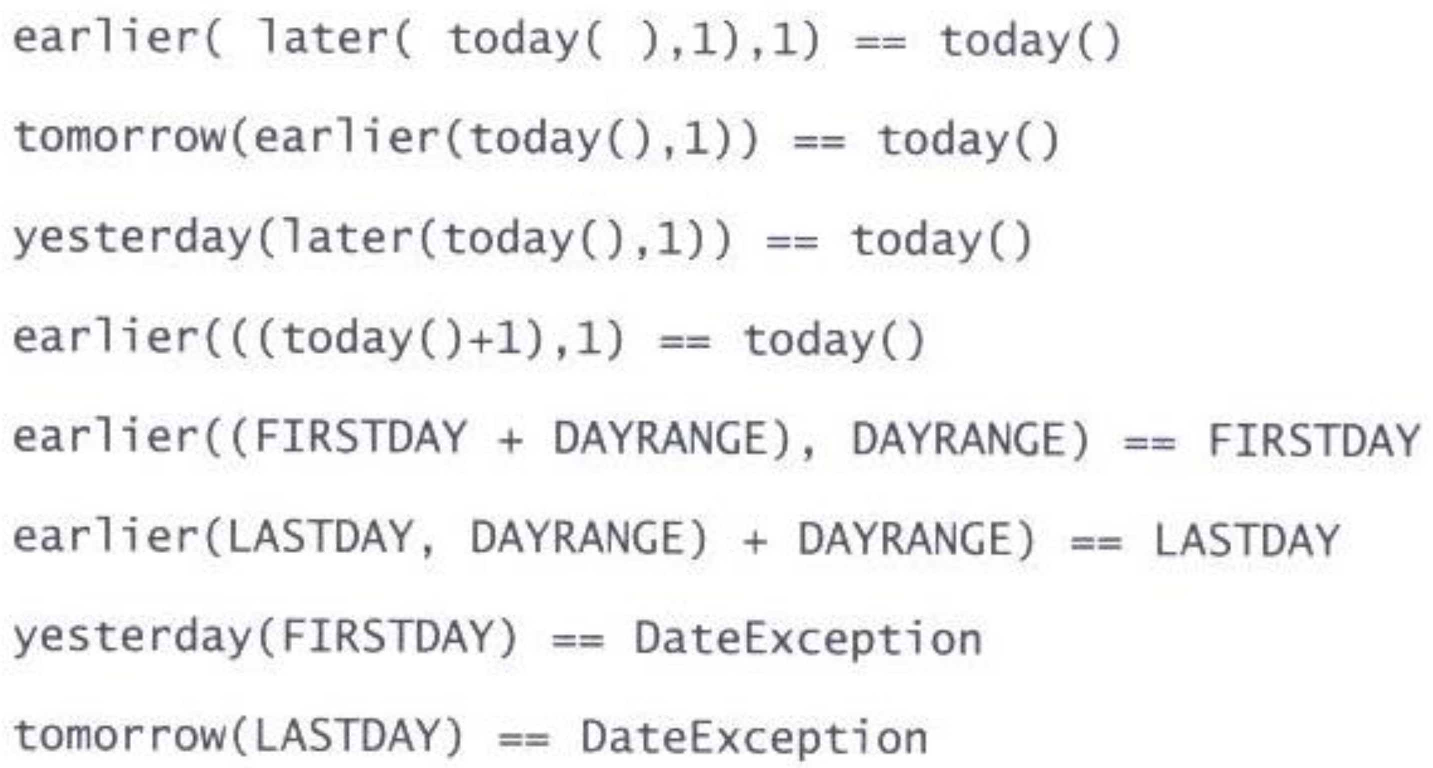


```
class Date {
   // ...

   Date tomorrow(Date aDay)
         {/* Returns one day after aDay */}

   Date yesterday(Date aDay)
         {/* Returns one before aDay */}

   Date today()
         {/* Returns todays date */}

   Date earlier(Date aDay, int diff)

         {/* Returns the date diff days before aDay /*}

   Date later(Date aDay, int diff)

         {/* Returns the date diff days after aDay /*}

   // ...
}
```

Reversible relationship: the date one day earlier than one day after today is today

assert (d.earlier(later(d.today(), 1),1)== d.today ();

Reversing oracles: reversible relationships



```
earlier( later( today( ),1),1) == today()

tomorrow(earlier(today(),1)) == today()

yesterday(later(today(),1)) == today()

earlier(((today()+1),1) == today()

earlier((FIRSTDAY + DAYRANGE), DAYRANGE) == FIRSTDAY

earlier(LASTDAY, DAYRANGE) + DAYRANGE) == LASTDAY

yesterday(FIRSTDAY) == DateException

tomorrow(LASTDAY) == DateException
```

Comparators

Automated comparison – a necessity (evaluation by inspection is not practical)

Categories of comparators

- System utilities (comparison of expected and actual results written to files – file comparison utilities)

- Smart comparators – greater control over comparison actions. COTS (commercially available of the shelf) tools, say File-Aid, DB Tester, Xdiff…

- Application-specific comparators

Oracles - evaluation

Judging oracles

Prespectification
- Solved example oracles
- Simulation oracles
- Approximation oracles

Gold standard oracles
- Trusted system oracle
- Parallel test oracle
- Voting oracle

Organic
- Smoke test
- Reversing oracles

	Low cost		High cost	
	narrow range	broad range	narrow range	broad range
High fidelity	•smoke test •reversing	•trusted system •parallel testing		•solved examples •voting
Low fidelity		judging	approximation	

Metamorphic testing (1)

Evaluation of correctness on a particular input;

DO NOT KNOW THE OUTPUT

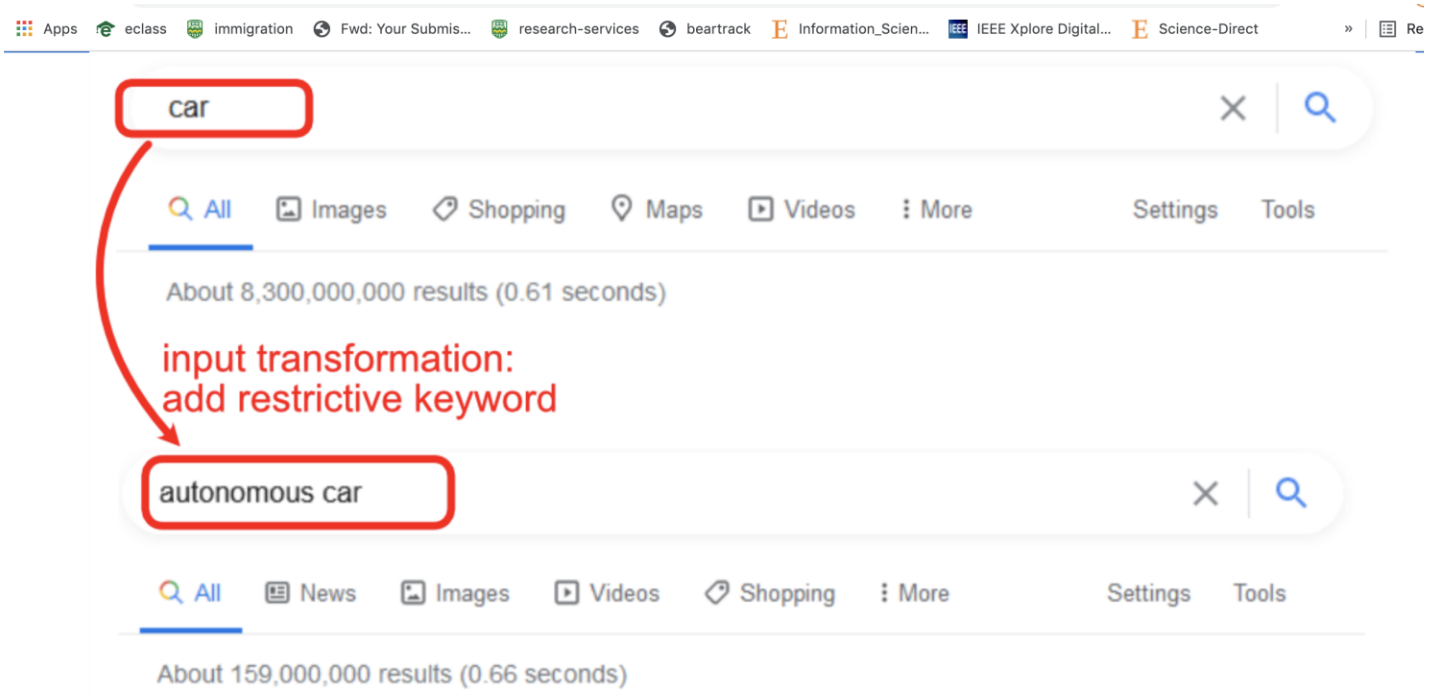
sin(x) for given x has to know the result, if the result is not known compute cos(x), use relationship $\sin^2(x) + \cos^2(x)=1$, $\sin(\pi-x)= \sin(x)$

Shortest path (SP) in a graph G SP(G; a, b)
SP(G; a, b) = SP(G; b, a)

Web search count of results produced for query *q,* Count(*q*)
Concatenation of *q* and *k*, *q*+*k*, Count(*q*+*k*) <Count(*q*)

Input transformation: for x generate new test case, compare the results of outputs for the pair of these inputs

Metamorphic testing-example



car

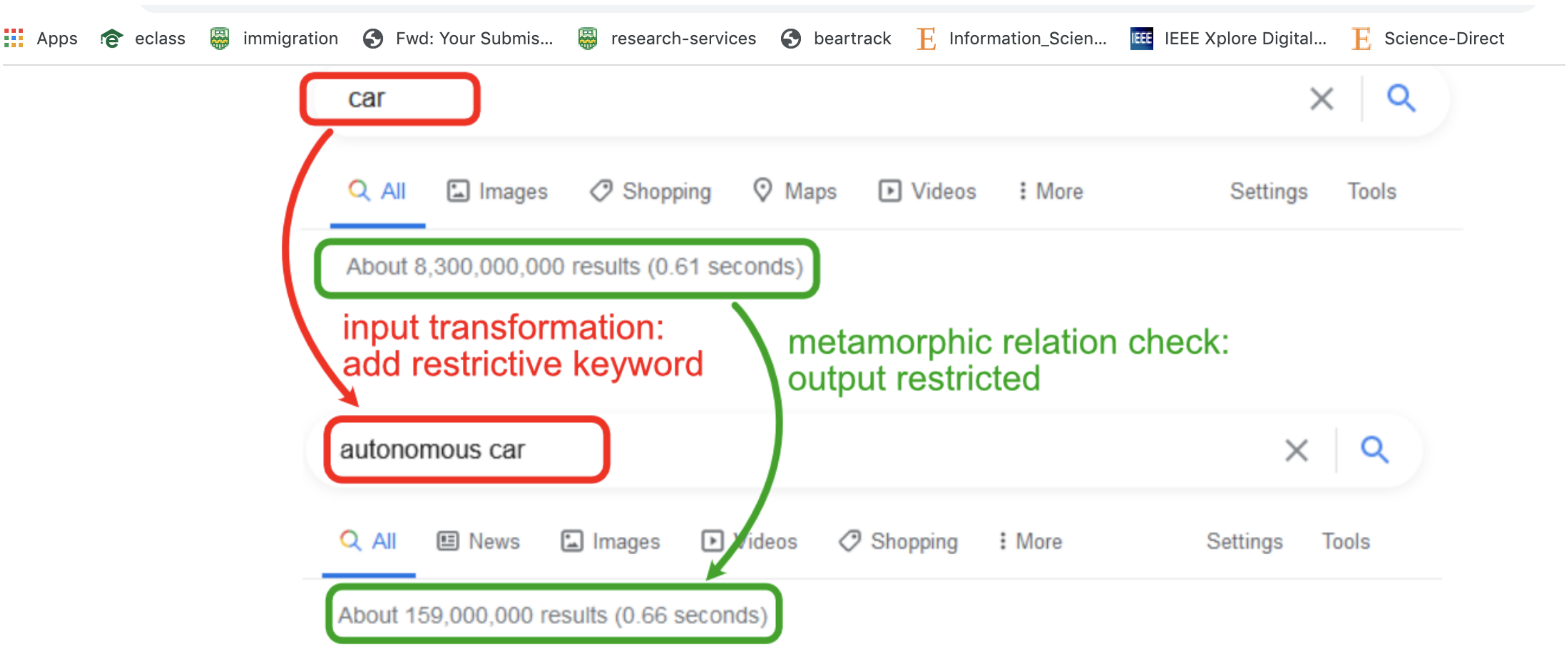
About 8,300,000,000 results (0.61 seconds)

input transformation:
add restrictive keyword

autonomous car

About 159,000,000 results (0.66 seconds)

Metamorphic testing -example



car

About 8,300,000,000 results (0.61 seconds)

input transformation:
add restrictive keyword

metamorphic relation check:
output restricted

autonomous car

About 159,000,000 results (0.66 seconds)

Example output (metamorphic) relation: More restricted query returns fewer results. Image by author.

Metamorphic testing - examples

computing

commutativity: f(a, b)= f(b, a)

permutation: f(a) = f(perm(x))

synonym replacement: result(“cheap flight”) = result(“inexpensive flight”)

Metamorphic testing (2)

Metamorphic relation

a relation for a function f - a relation among series of inputs x_1, x_2, ..., x_n, n>1 and their corresponding output values $f(x_1)$, $f(x_2)$,..., $f(x_n)$.

sin(x)

$R =\{(x_1, x_2, \sin(x_1), \sin(x_2)| x_2 = \pi-x_1, \sin(x_1) = \sin(x_2)\}$

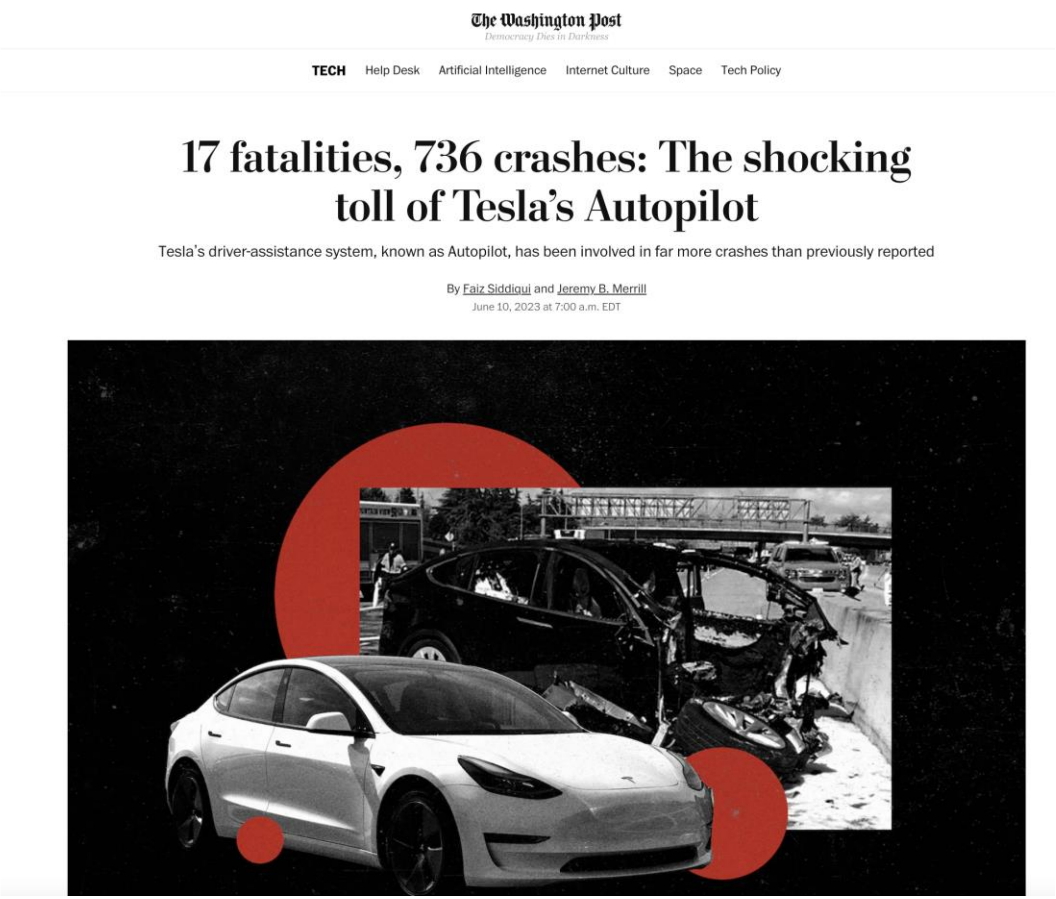
$(x_1, \sin(x_1)$ - source test case
$(x_2, \sin(x_2))$ - follow-up test case

$(x_1\ x_2=f(x_1)) \qquad x_1\ R\ f(x_1)$

If the relation R does not hold then the fault has been detected

The process could be automated.

Vision system for autonomous vehicles: metamorphic relations



The Washington Post

Democracy Dies in Darkness

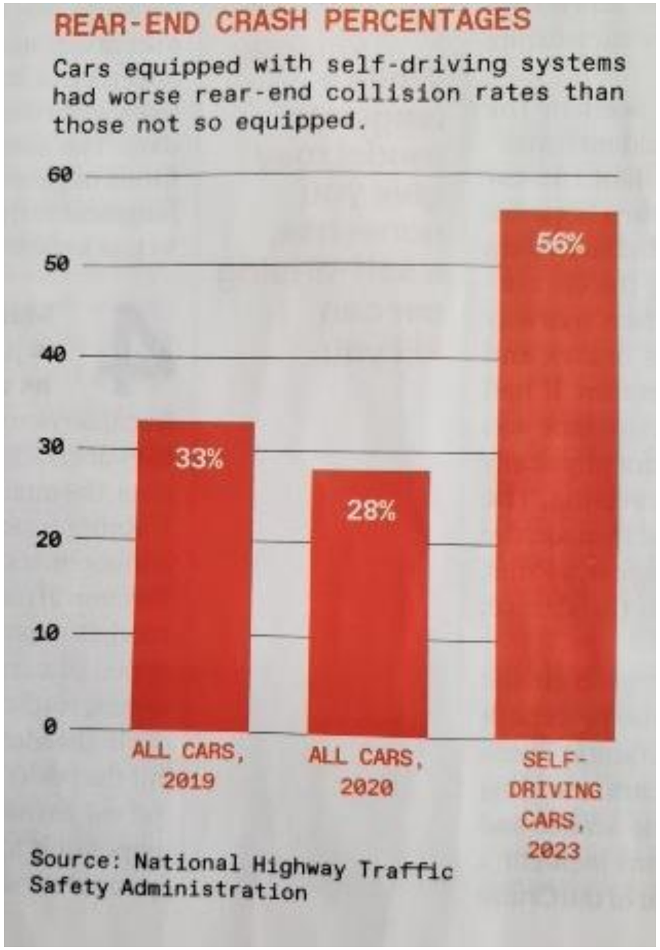
TECH Help Desk Artificial Intelligence Internet Culture Space Tech Policy

17 fatalities, 736 crashes: The shocking toll of Tesla's Autopilot

Tesla's driver-assistance system, known as Autopilot, has been involved in far more crashes than previously reported

By Faiz Siddiqui and Jeremy B. Merrill

June 10, 2023 at 7:00 a.m. EDT



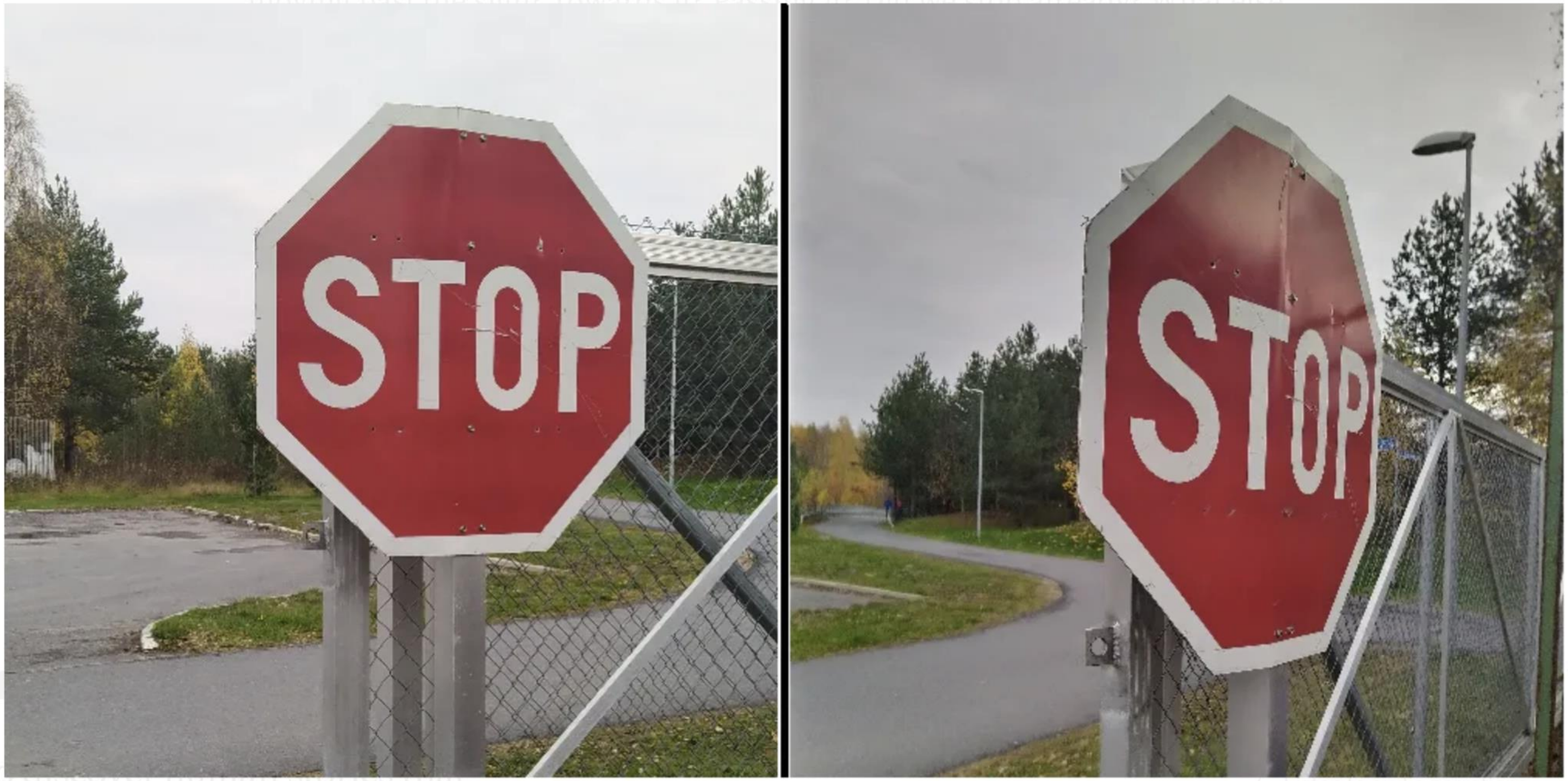
REAR-END CRASH PERCENTAGES

Cars equipped with self-driving systems had worse rear-end collision rates than those not so equipped.

Source: National Highway Traffic Safety Administration

Vision system for autonomous vehicles: metamorphic relations

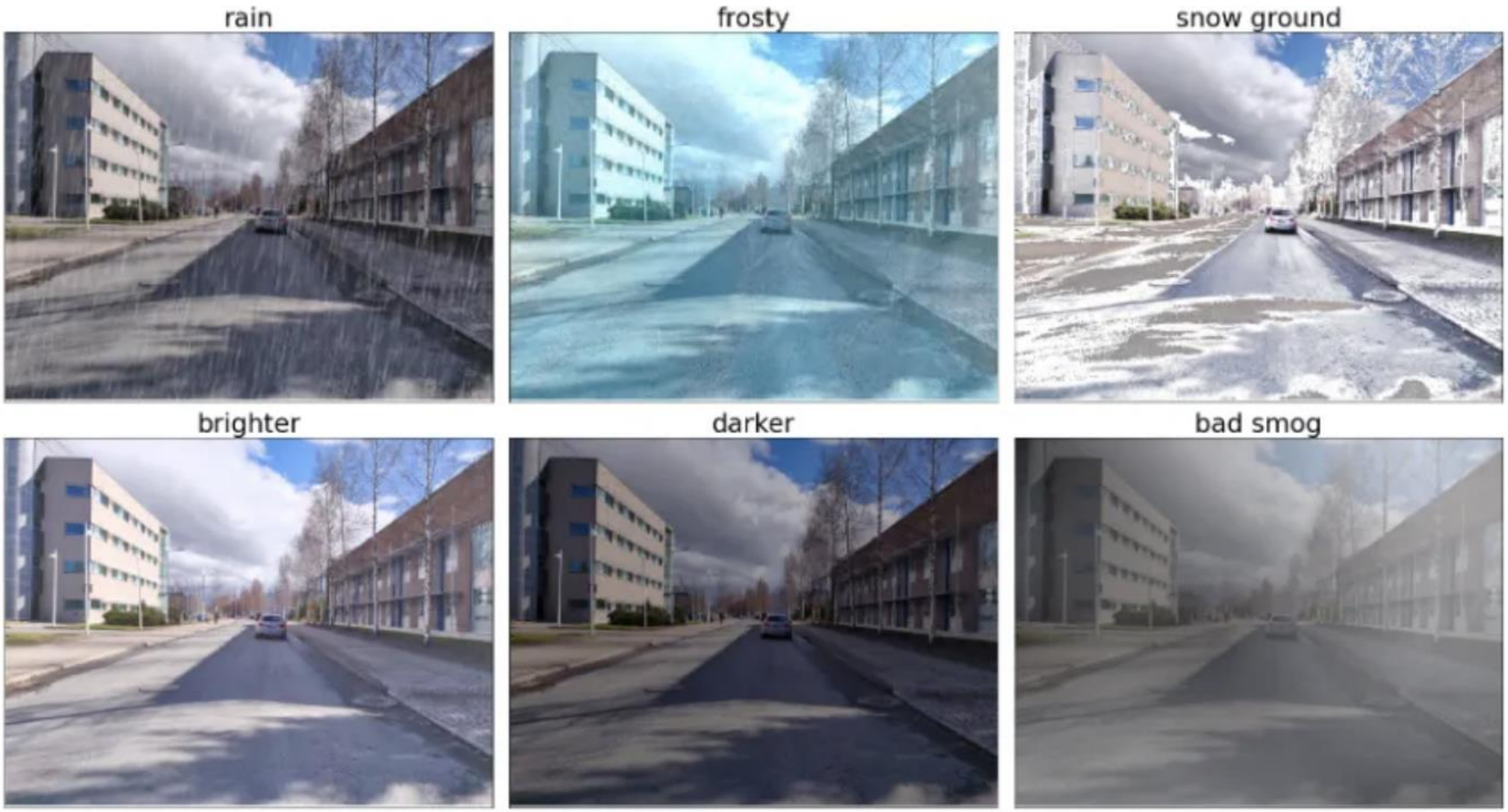
moving past the sign? Towards it? Passing it? Did we stop already? What else



o discover human stories that
our understanding of the world.

✓ Listen to audio narrations

Vision system for autonomous vehicles: metamorphic relations



rain

frosty

snow ground

brighter

darker

bad smog

Vision system for autonomous vehicles: metamorphic relations

Translation Invariance

Scale Invariance

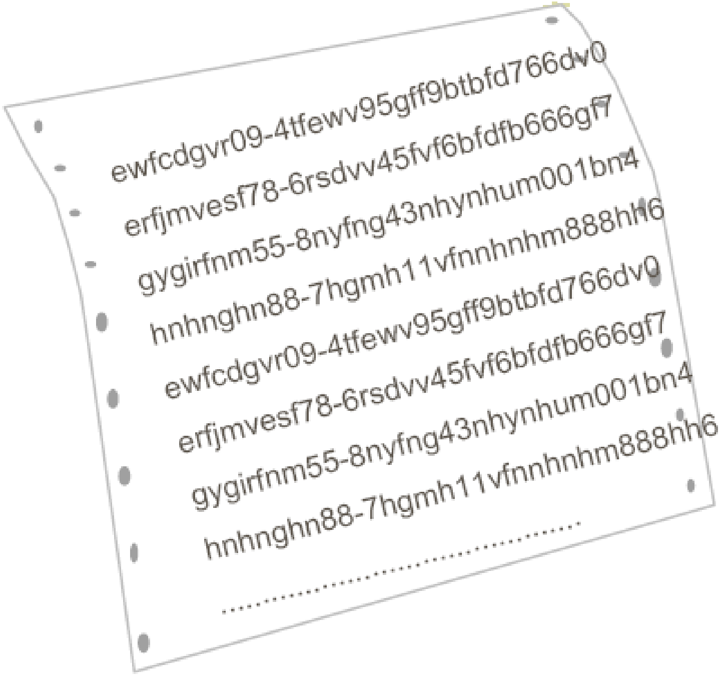
Rotation Invariance

Brightness and contrast Invariance

Noise Resilience

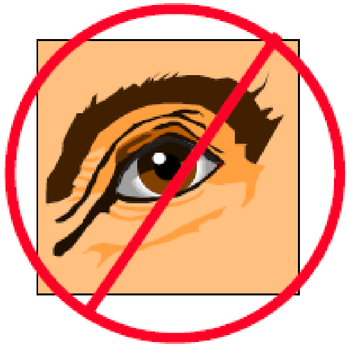
Color Invariance

Concluding Comment



```
ewfcdgvr09-4tfewv95gff9btbfd766dv0
erfjmvesf78-6rsdvv45fvf6bfdfb666gf7
gygirfnm55-8nyfng43nhynhum001bn4
hnhnghn88-7hgmh11vfnnhnhm888hh6
ewfcdgvr09-4tfewv95gff9btbfd766dv0
erfjmvesf78-6rsdvv45fvf6bfdfb666gf7
gygirfnm55-8nyfng43nhynhum001bn4
hnhnghn88-7hgmh11vfnnhnhm888hh6
......................................................
```

Test suites may contain hundreds or thousands of test cases



The amount of data to be checked cannot be handled by human beings

Automated oracles are essential!