Causes and Effects
Andreas Zeller
double bug(double z[], int n) {
    int i, j;

    i = 0;
    for (j = 0; j < n; j++) {
        i = i + j + 1;
        z[i] = z[i] * (z[0] + 1.0);
    }
    return z[n];
}
Where is the error which causes this failure?
Locating Errors

An error is a deviation from what is correct, right, or true:

- **Input** ("The URL must be well-formed")
- **Variables** ("link is zero")
- **Statements** ("even(2) must return true")

How do we know one of these is correct?

How can we say “The defect is here”? 
Locating Causes

An aspect of the execution *causes* a failure if it can be altered such that the failure no longer occurs:

- *Input* ("11 14")
- *Variables* ("argc = 2")
- *Statements* ("Line 37")

Note that a cause need not be an error!
Causality

The notion of *causality* is deeply linked to fundamental questions of philosophy:

- What is it that makes things happen?
- Can we predict the future from causes?
- If everything has a cause, what is the ultimate cause of events in the past?
Aristotle
(384–322 BC)
Aristotle on Causality

Aristotle suggested four types of causes:

• The *material* of which things come
• The *form* which things have when they are perfected
• The *moving* cause or actual agent
• The *purpose* or *function* of such things
Example

Creating a silver chalice for a religious ceremony

- Material cause – the silver
- Formal cause – the design of the chalice
- Efficient cause – the silversmith
- Final cause – the religious ceremony
William of Ockham
(1288–1349)
Ockham on Causality

• The only way in which we can establish any causal connection between one thing and another is the observation that when one of these occurs, the other also occurs at the same time and at or near the same place.

• This is the only way to establish causality
David Hume
(1711–1776)
Hume on Causality

• When we see that two events always occur together, we tend to form an expectation that when the first occurs, the second will soon follow.

• This constant conjunction and the expectation thereof is all that we can know of causation, and all that our idea of causation can amount to.
Causality as Illusion

- Just because the sun has risen every day since the beginning of the Earth does not mean that it will rise again tomorrow.

- Bertrand Russell: “causation = superstition”
Counterfactuals

• We may define a *cause* to be an object followed by another, and where all the objects, similar to the first, are followed by objects similar to the second. Or, in other words, where, *if the first object had not been, the second never had existed.* (Hume, 1748)

• Hume never explored this alternative
Causality

Actual world

Effect does occur

Effect does not occur

Alternate world

Causes
double bug(double z[], int n) {
    int i, j;
    i = 0;
    for (j = 0; j < n; j++) {
        i = i + j + 1;
        z[i] = z[i] * (z[0] + 1.0);
    }
    return z[n];
}
double bug(double z[], int n) {
    int i, j;

    i = 0;
    for (j = 0; j < n; j++) {
        i = i + j + 1;
        z[i] = z[i] * (z[0] + 1.0);
    }
    return z[n];
}
Causes as Differences

Actual world

empty.c: GCC works fine

bug.c: GCC crashes

Alternate world

Cause: bug.c
More possible causes

<table>
<thead>
<tr>
<th>GCC code</th>
<th>invocation</th>
<th>me</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux</td>
<td>electricity</td>
<td>oxygen</td>
</tr>
</tbody>
</table>
Lewis on Causation

- $C \rightarrow E$ means “If $C$ had been the case, $E$ would have been the case”
- $C$ causes $E$ if $C \rightarrow E$ and $\neg C \rightarrow \neg E$ hold.
- $C \rightarrow E$ holds if some $C$-world where $E$ holds is closer to the actual world than is any $C$-world where $E$ does not hold.
C \iff E holds if some C-world where E holds is closer to the actual world than is any C-world where E does not hold.

\begin{itemize}
  \item A world with an alternate GCC input is closer than a world without oxygen
  \item A world with GCC fixed may be closer than a world with an alternate GCC input
\end{itemize}
Actual Causes

“The” cause (actual cause) is a minimal difference
double bug(double z[], int n) {
    int i, j;
    i = 0;
    for (j = 0; j < n; j++) {
        i = i + j + 1;
        z[i] = z[i] * (z[0] + 1.0);
    }
    return z[n];
}
Isolating Causes

double bug(double z[], int n) {
    int i, j;

    i = 0;
    for (j = 0; j < n; j++) {
        i = i + j + 1;
        z[i] = z[i] * (z[0] + 1.0);
    }
    return z[n];
}
Isolating Causes

double bug(double z[], int n) {
    int i, j;
    i = 0;
    for (j = 0; j < n; j++) {
        i = i + j + 1;
        z[i] = z[i] * (z[0] + 1.0);
    }
    return z[n];
}
double bug(double z[], int n) {
    int i, j;

    i = 0;
    for (j = 0; j < n; j++) {
        i = i + j + 1;
        z[i] = z[i] * (z[0] + 1.0);
    }
    return z[n];
}

Actual cause narrowed down
double bug(double z[], int n) {
    int i, j;
    i = 0;
    for (j = 0; j < n; j++) {
        i = i + j + 1;
        z[i] = z[i] * (z[0] + 1.0);
    }
    return z[n];
}
Isolating Causes

double bug(double z[], int n) {
    int i, j;

    i = 0;
    for (j = 0; j < n; j++) {
        i = i + j + 1;
        z[i] = z[i] * (z[0] + 1.0);
    }

    return z[n];
}
Isolating Causes

double bug(double z[], int n) {
  int i, j;

  i = 0;
  for (j = 0; j < n; j++) {
    i = i + j + 1;
    z[i] = z[i] * (z[0] + 1.0);
  }

  return z[n];
}

Actual cause of the GCC crash
Isolating Causes

Actual world

Alternate world

Mixed world

Test
Isolating Causes

Actual world → "+ 1.0" ← Alternate world

Test

Mixed world

✔

✘
Search Space

The choice of an *initial set of differences* determines the search space for causes:

- the input (data, configuration, …)
- the program state
- the program code

Sets a *common context* between worlds
## Search Space

<table>
<thead>
<tr>
<th>Input</th>
<th>State</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS</td>
<td>Compiler</td>
<td>Processor</td>
</tr>
<tr>
<td>FBI</td>
<td>E.T.</td>
<td>Them!</td>
</tr>
</tbody>
</table>
Ockham’s Razor

• Whenever you have competing theories for how some effect comes to be, *pick the simplest.*
Ockham’s Razor

In our context:

• Whenever you have the choice between multiple causes, pick the one whose alternate world is closer.
## Search Space

<table>
<thead>
<tr>
<th>Input</th>
<th>State</th>
<th>Code</th>
<th>Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS</td>
<td>Compiler</td>
<td>Processor</td>
<td>close</td>
</tr>
<tr>
<td>FBI</td>
<td>E.T.</td>
<td>Them!</td>
<td>far away</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>far out</td>
</tr>
</tbody>
</table>
Hanlon’s Razor

- Never explain by malice which is adequately explained by stupidity
Verifying Causes

Do we know the configuration in .psharprc causes the failure?

```bash
$ ./psharp db.p#
.psharprc: 37: no such interpreter
.psharprc: 37: bailing out
Segmentation fault
```
Causes and Effects

To prove causality, one must show that

- the effect occurs when the cause occurs
- the effect does not occur when the cause does not.

This is the only way to prove causality
Verifying Causes

So it wasn’t the configuration after all

$ mv ~/.psharprc ~/.psharprc.orig
$ ./psharp db.p#
Segmentation fault
Verifying Causes

Avoid *post hoc ergo propter hoc* fallacies

```
$ ./psharp db.p#
.psharprc: 37: no such interpreter
.psharprc: 37: bailing out
Segmentation fault
```
Verifying Causes

```c
a = compute_value();
printf("a = %d\n", a);
```

\[ a = 0 \]
Is variable \( a \) zero?

\[
\begin{align*}
  a &= \text{compute\_value}(); \\
  a &= 1; \\
  \text{printf}("a = \%d
", a); \\
  a &= 0
\end{align*}
\]
What’s going on?

double a;
a = compute_value();
a = 1;
printf("a = %d\n", a);

a = 0
What’s going on?

double a;
a = compute_value();
printf("a = %f\n", a);

a = 3.14...
What’s going on?

double a;
a = compute_value();
printf("a = %f\n", a);

We have isolated the format "%d"
as the actual failure cause
Preemption

Billy and Suzy throw rocks at a bottle. Suzy throws first so that her rock arrives first and shatters the glass. Without Suzy's throw, Billy's throw would have shattered the bottle.

• Does Suzy’s throw cause the shattering?
Alteration

• C influences E if C can be altered to C’ such that E’ occurs instead of E (Lewis; 1999)

• If Suzy had not thrown the stone, the bottle would have shattered in a different manner

• Therefore, Suzy’s throw influenced and caused the original shattering
What’s the Failure?

• Every failure has some aspects that we consider relevant

• This choice influences the search for causes

• If the entire state of the program is part of the failure, we get very detailed causes

• If just one aspect is relevant, we get simpler causes – sometimes too simple
Concepts

A *cause* is an event preceding another event (the *effect*) without which the effect would not have occurred.

A cause can be seen as a *difference* between a world where the effect occurs and a world where it does not.

An *actual* cause means a *minimal difference*.
If $C$ is a cause and $E$ is its effect, then $C$ must precede $E$. 

- yes 
- no
If $C$ is a circumstance that causes a failure, then it is possible to change $C$ such that the failure no longer occurs.

- [x] yes
- [ ] no
If some cause $C$ is an actual cause, then altering $C$ induces the smallest possible difference in the effect.

☐ yes  ❌ no
Every failure cause implies a possible fix.

✓ yes  □ no
Quiz

For every failure, there is exactly one actual cause.

☐ yes  ☒ no
Quiz

For every defect, there is exactly one correction.

☑ yes  ☐ no
A failure cause can be determined without executing the program.

[ ] yes  [x] no
Quiz

A failure is the difference to the closest possible world in which the cause does not occur.

☐ yes  ❌ no
If I observe two runs (one passing, one failing) with a minimal difference in input, then I have found an actual failure cause.

- [x] yes
- [ ] no
A minimal and successful correction proves that the altered code was the actual failure cause.
Increasing the common context between the possible worlds results in smaller causes.

- [ ] yes
- [x] no