Causality

- When a test case fails we start debugging
- We assume that the fault (what we’re really after) *causes* the failure
  - Remember RIP (Reachability, Infection, Propagation)?
- What do we mean when we say that
  - “A causes B”?

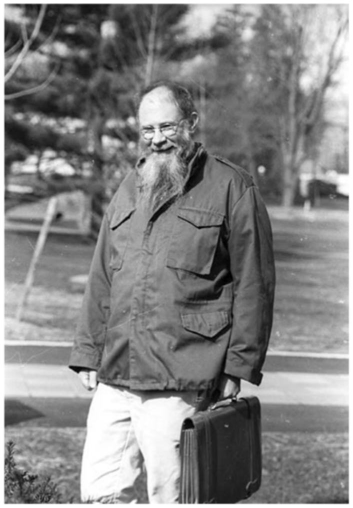
Causality

- *We don’t know*
- Though it is central to everyday life – and to the aims of science
  - A real understanding of causality eludes us to this day
  - Still no non-controversial way to answer the question “does A cause B”?

Causality

- Philosophy of causality is a fairly active area, back to Aristotle, and (more modern approaches) Hume

  - General agreement that a cause is something that “makes a difference” – if the cause had not been, then the effect wouldn’t have been
  - One theory that is rather popular with computer scientists is David Lewis’ *counterfactual* approach
    - Probably because it (and probabilistic or statistical approaches) are amenable to mathematical treatment and automation



Causality (According to Lewis)

- For Lewis (roughly – I'm conflating his *counterfactual dependency* and *causal dependency*)
  - A causes B (in world *w)* iff
  - In all *possible worlds* that are *maximally similar* to *w*, and in which A does not take place, B also does not take place

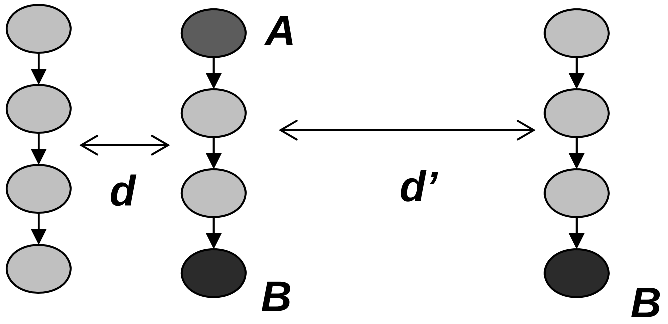
Causality (According to Lewis)

- Causality does not depend on
  - B being *impossible* without A
  - Seems reasonable: we don’t, when asking “Was Larry slipping on the banana peel causally dependent on Curly dropping it?” consider worlds in which new circumstances (Moe dropping a banana peel) are introduced

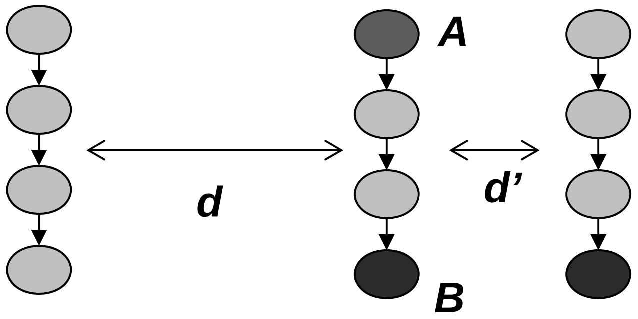
Causality (According to Lewis)

- Many objections to Lewis in the literature
  - e.g. cause precedes the event in time seems to not be required by his approach
- One is not a problem for our purposes
  - Distance metrics (how similar is world *w* to world *w'*) are problematic for “worlds”
    - Counterfactuals are tricky
  - Not a problem for *program executions*
    - May be details to handle, but no one has in-principle objections to asking how similar two program executions are
    - Or philosophical problems with multiple executions (no run is “privileged by actuality”)

Causality (According to Lewis)



Yes! d < d'



No. d > d'

Did A cause B in this program execution?

Formally

A predicate e is causally dependent on a predicate c in an execution a iff:

1. $c(a) \wedge e(a)$
2. $\exists b\ .\ (\neg c(b) \wedge \neg e(b) \wedge$
   $(\forall b'\ .\ (\neg c(b') \wedge e(b')) \Rightarrow (d(a, b) < d(a, b'))))$

What does this have to do with automated debugging??

- A **fault** is an incorrect part of a program
- In a failing test case, some fault is reached and executes
  - *Causing* the state of the program to be corrupted (**error**)
  - This incorrect state is propagated through the program (propagation is a series of “A causes B”s)
  - Finally, bad state is observable as a **failure** – *caused* by the **fault**

Fault Localization

- **Fault localization**, then, is:
  - An effort to automatically find (one of the) causes of an observable failure
  - It is inherently difficult because there are many causes of the failure that are not the fault
    - We don’t mind seeing the chain of cause and effect reaching back to the fault
    - But the fact that we reached the fault at all is also a cause!