Causality in Concurrent Systems

F. Russo

Vrije Universiteit Brussel

Belgium

S.Crafa

Università di Padova

Italy

HaPoC 31 October 2013, Paris

Causality in Concurrent Systems

Vrije U

software, hardware or even physical systems
where
sets of activities **run in parallel**
with
possible occasional **interactions**

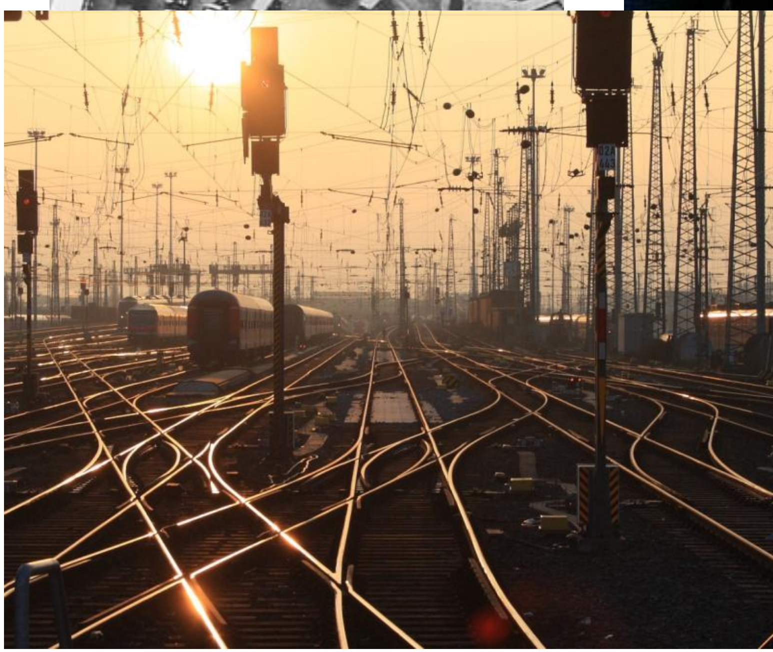
HaPoC *31 October 2013, Paris*

Concurrent Systems

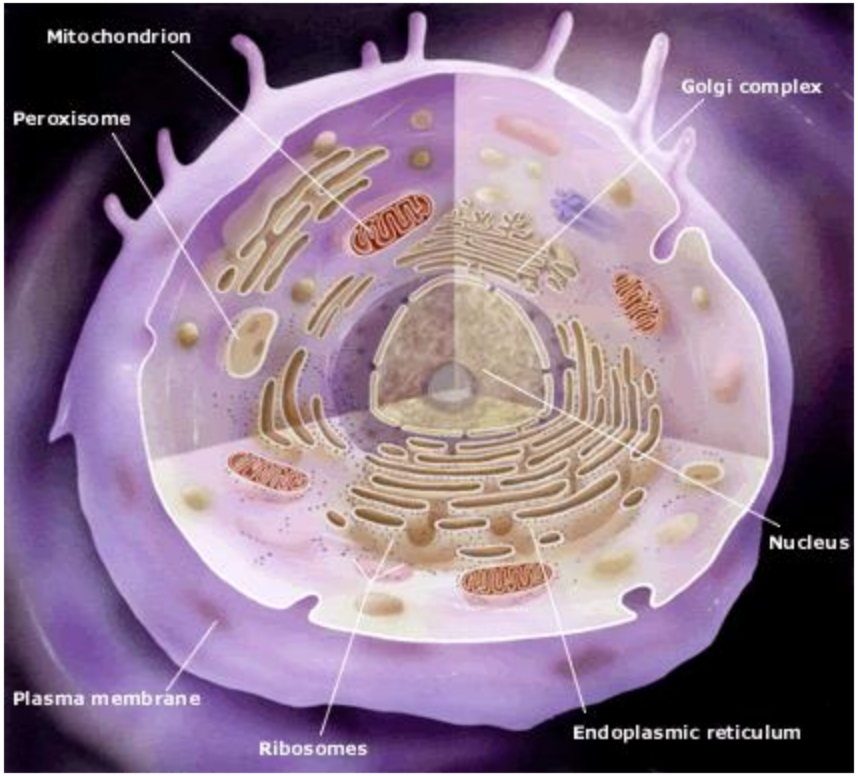
Concurrent Systems

Concurrent Systems

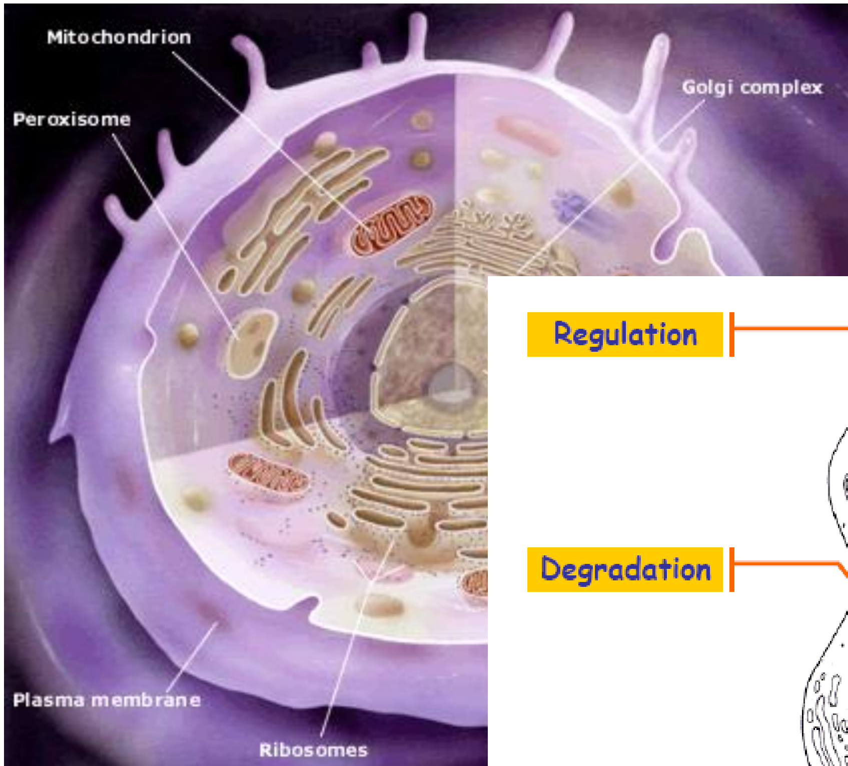
Concurrent Systems

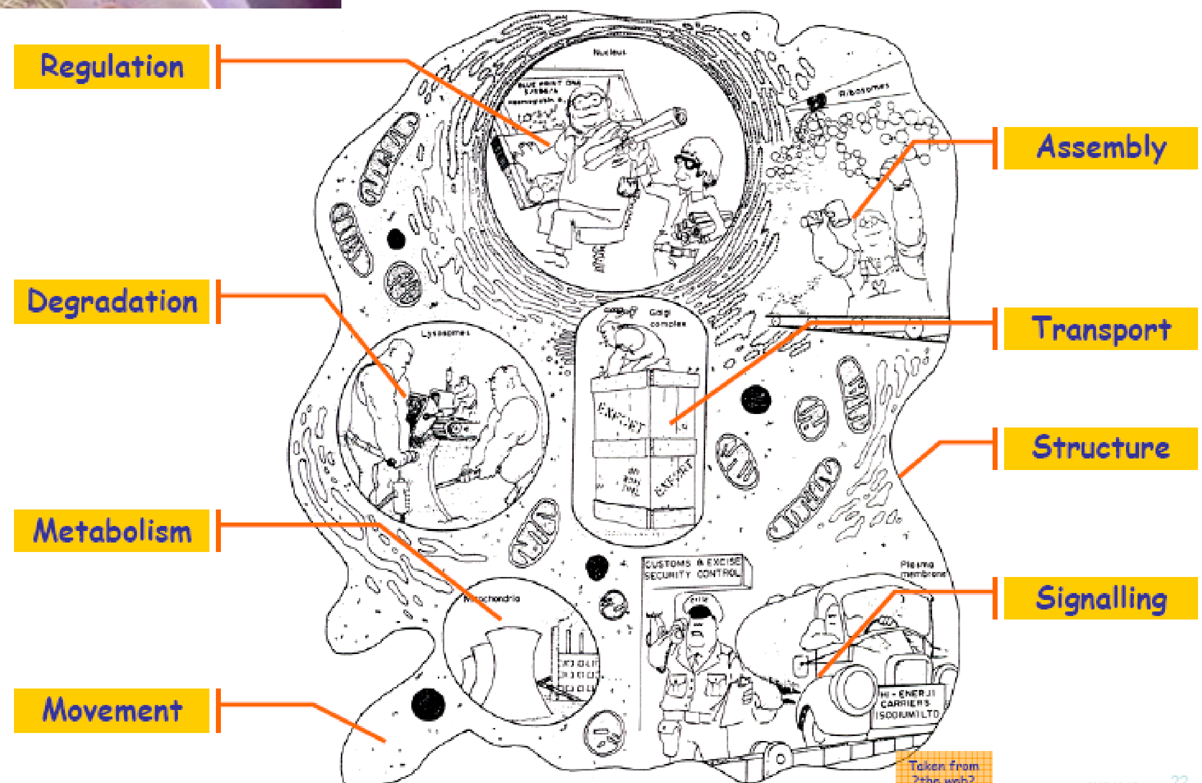


Concurrent Systems



Concurrent Systems





Overview

- Concurrent Systems
  - ***How to deal with such a complexity?***
  - CS offers tools: Formal/precise, expressive/general, simple/tractable

Overview

- Concurrent Systems
  - ***How to deal with such a complexity?***
  - CS offers tools: Formal/precise, expressive/general, simple/tractable

Formal language

to *specify* the system

- Java programming language, …
- DSL for concurrent hardware, system biology, …
- process algebras

Overview

- Concurrent Systems
  - *How to deal with such a complexity?*
  - CS offers tools: Formal/precise, expressive/general, simple/tractable

- Interleaving models (1 action at a time)
- ***true-concurrent***, ***causal models*** take the notion of concurrency/causality as fundamental

Operational model
to describe all system *behaviors*

Formal language
to *specify* the system

Overview

- Concurrent Systems
  - *How to deal with such c*
  - CS offers tools: Formal/|

what causality means ?

w.r.t. traditional debates in philosophy of causality:

production and mechanisms, independence, causation by omission

ble

- Interleaving models (1 action at a time)
- ***true-concurrent***, ***causal models*** take the notion of concurrency/causality as fundamental

Operational model

to describe all system *behaviors*

Formal language

to *specify* the system

how causality is formalized (in PESs)?

Overview

- Concurrent Systems
  - *How to deal with such a complexity?*
  - CS offers tools: Formal/precise, expressive/general, simple/tractable

Analysis techniques
to reason/prove system *properties*

- ***before*** system exec.: static analysis
- ***during*** system exec.: dynamic analysis / execution profiling
- ***after*** an actual exec.: trace analysis / fault diagnosis (examining causal history of error occurrence)

Operational model
to describe all system *behaviors*

Formal language
to *specify* the system

Overview

- Concurrent Systems
  - ***How to deal with such a complexity?***
  - CS offers tools: Formal/precise, expressive/general, simple/tractable

Analysis techniques
to reason/prove system *properties*

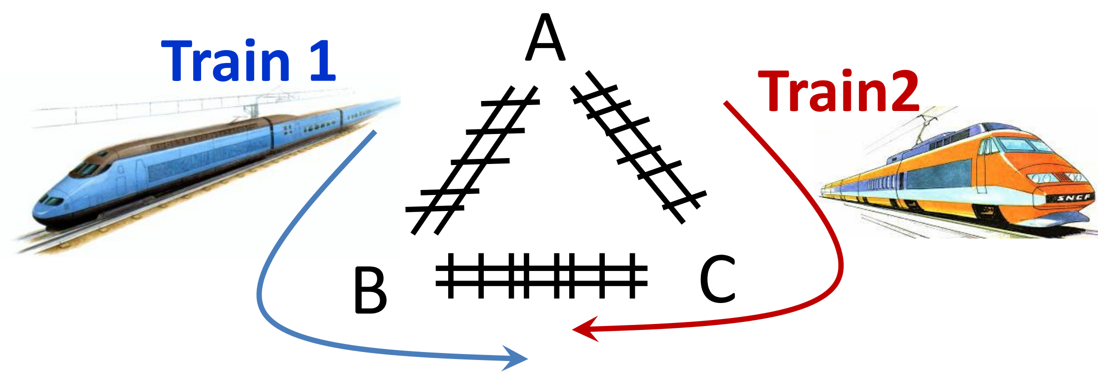
- ***before*** system exec.: static analysis
- ***during*** system exec.: dynamic analysis / execution profiling
- ***after*** an actual exec.: trace analysis / fault diagnosis (examining causal history of error occurrence)

Operational model
to describe all system *behaviors*

Formal language
to *specify* the system

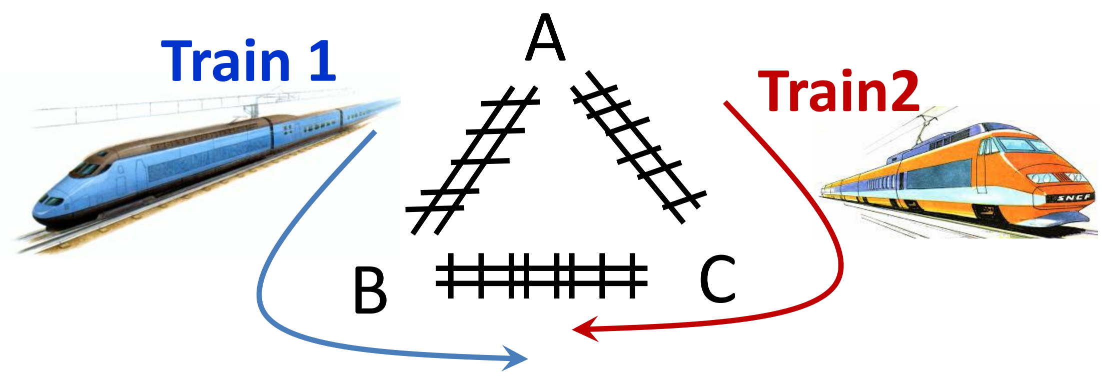
counterfactual reasoning
as an example of causal reasoning:
c. validation and refutation using the theory of ***N. Rescher***

Example: a railway system



- each pair of stations connected by a single track
- Train1 and Train2 **move concurrently** (at possibly different speed) between A-B and A-C.
- The transit between B and C must be regulated: **no collision!**
- between B and C must be **in mutual exclusion**

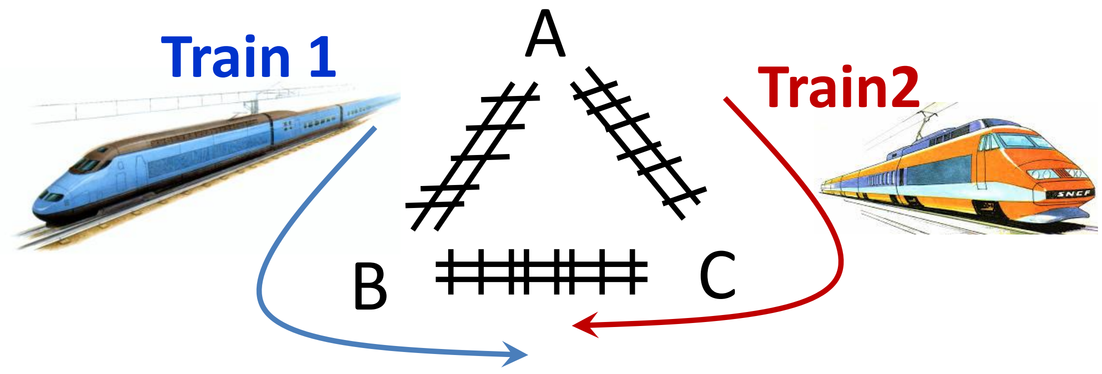
Example: a railway system



The presence of Train1 at C ***depends*** on its previous presence at B

- Train1 and Train2 **move concurrently** (at possibly different speed) between A-B and A-C.
- The transit between B and C must be regulated: **no collision!**
- between B and C must be **in mutual exclusion**

Example: a railway system

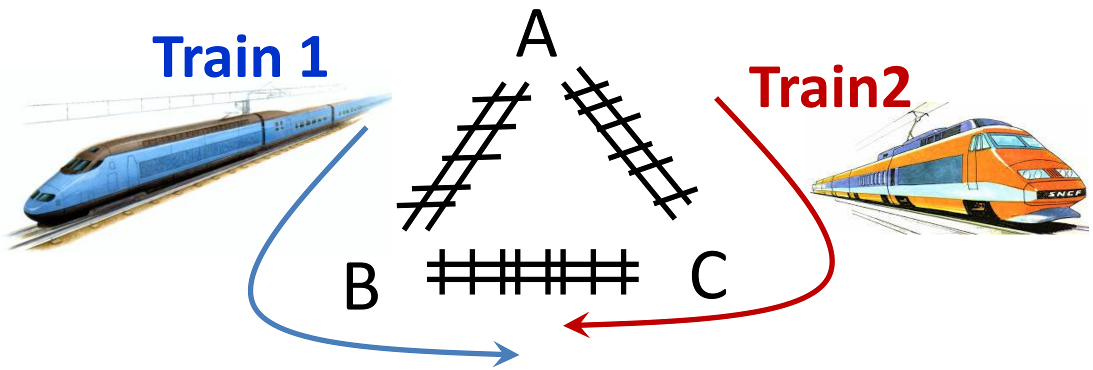


The presence of Train1 at C ***depends*** on its previous presence at B

Train1 on the track AB and Train2 on the track AC are ***concurrent*** activities: they can take place in **any order**, or **at the same time** as well

- The transit between B and C must be regulated: **no collision!**
- between B and C must be **in mutual exclusion**

Example: a railway system

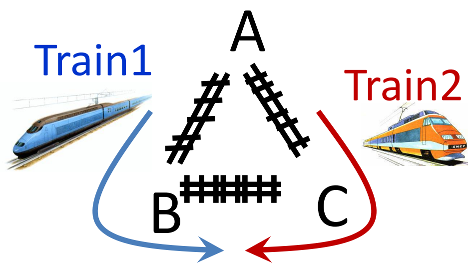


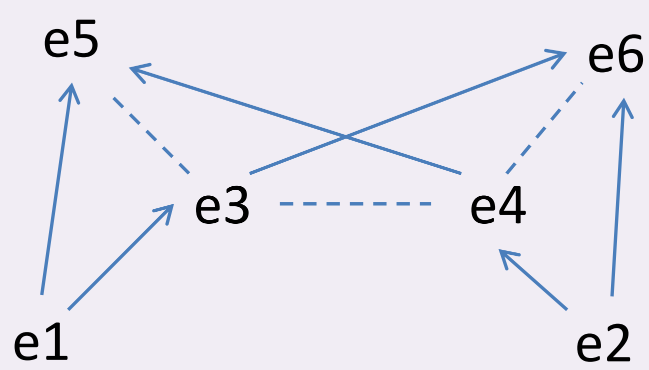
The presence of Train1 at C ***depends*** on its previous presence at B

Train1 on the track AB and Train2 on the track AC are ***concurrent*** activities: they can take place in **any order**, or **at the same time** as well

The usage of track BC by Train1 is ***in conflict*** with the usage of the same track by Train2: **any of the two, but not both**

The railway system as a PES

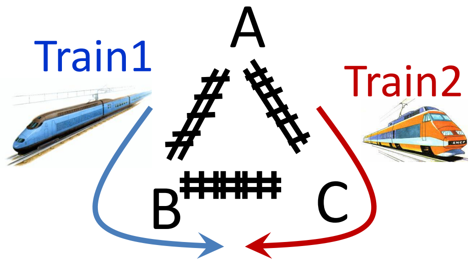


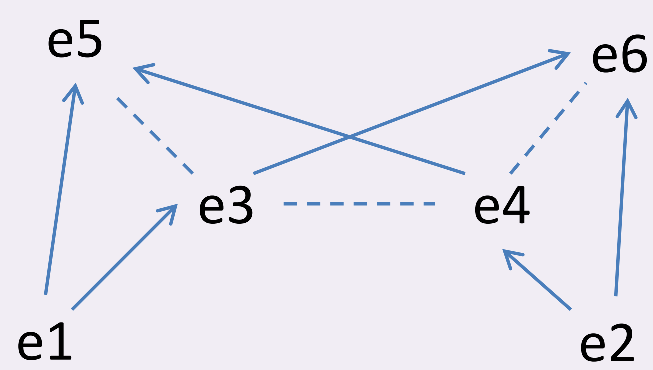


λ(e1) = Train1 in tract AB
λ(e2) = Train2 in track AC
λ(e3) = Train1 in track BC
λ(e4) = Train2 in track BC
λ(e5) = Train1 in track BC
λ(e6) = Train2 in track BC

- **Labeled Prime Event Structure** <E,<,#,λ>
  - E is a set of events e (event=a step of computation)
  - λ(e) action associated to the occurrence of e
  - < a **partial order** representing **the causal relation** between events:
    - e1<e3 ***e1 is a cause of e3***
  - # irreflexive and symmetric relation called **conflict**:
    - e3#e4 two alternative behaviors
  - axiom: the conflict is hereditary: if e<e’ and e#e’’ then e’#e’’

The railway system as a PES





λ(e1) = Train1 in tract AB

λ(e2) = Train2 in track AC

λ(e3) = Train1 in track BC

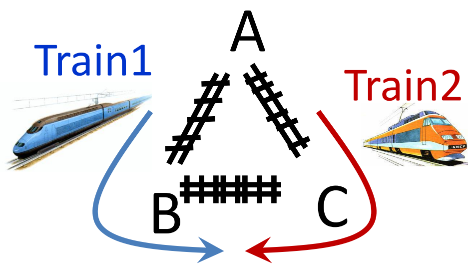
λ(e4) = Train2 in track BC

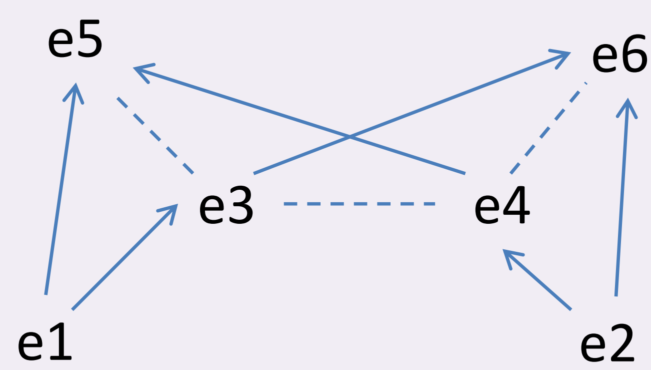
λ(e5) = Train1 in track BC

λ(e6) = Train2 in track BC

- **Labeled Prime Event Structure** (flow e.s., asymetric conflict,...)
- Petri nets (multiple tokens, open nets, ...)
- generalized Labeled Transition Systems
- (unstable) configuration structures
- causal trees
- ...

The railway system as a PES





λ(e1) = Train1 in tract AB

λ(e2) = Train2 in track AC

λ(e3) = Train1 in track BC

λ(e4) = Train2 in track BC

λ(e5) = Train1 in track BC

λ(e6) = Train2 in track BC

- **Labeled Prime Event Structure** (flow e.s., asymetric conflict,…)
- Petri nets (multiple token
- generalized Labeled Tra
- (unstable) configuration s
- causal trees
- …

true-concurrent models

where appears ***causal talking***

??

The meaning of causality in PESs

- **causality is a primitive relation**:

  e1 causally depends on e2 iff it has been so defined

The meaning of causality in PESs

- **causality is a primitive relation**:

  e1 causally depends on e2 iff it has been so defined

- These models are intended to be used
  - not for causal ***discovery***
  - but for (formal and automatic) ***reasoning on top*** of causal relations,

    e.g. **prove** that 'at any time A depends on B and it is concurrent with C'

The meaning of causality in PESs

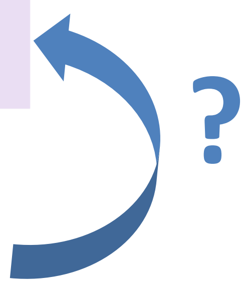
- **causality is a primitive relation**:

  e1 causally depends on e2 iff it has been so defined

- These models are intended to be used
  - not for causal ***discovery***
  - but for (formal and automatic) ***reasoning on top*** of causal relations,

    e.g. **<u>prove</u>** that 'at any time A depends on B and it is concurrent with C'

Operational model
to describe all system *behaviors*



Formal language
to *specify* the system

how to define the causal relation <
so that
the resulting PES **agrees**
with the system behaviors

The meaning of causality in PESs

- instruction **n+1** causally depends on instruction **n**
  - ok in sequential code
  - but: instruction n = "*execute this in parallel with the following*"
  - but: runtime reorders instructions to optimize exec.

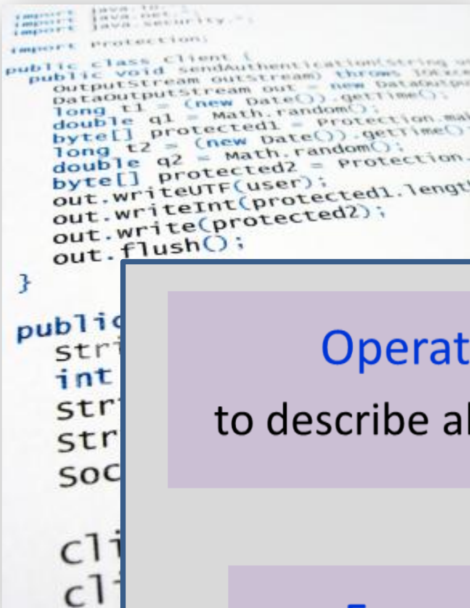
- many approaches, research issue
- no causal discovery
- debate about how to define a precedence relation

The meaning of causality in PESs

- instruction **n+1** causally depends on instruction **n**
  - ok in sequential code
  - but: instruction n = "*execute this in parallel with the following*"
  - but: runtime reorders instructions to optimize exec.

- the (Java) **Memory Model** *defines* a partial ordering on program instructions, *the happens-before relation*
  - runtime reordering must respect the HB-relation
  - new parallel hardware (multicore CPUs, GPUs) requires new memory model

The meaning of causality in PESs



- instruction **n+1** causally depends on instruction **n**
  - <u>ok</u> in sequential code
  - <u>but</u>: instruction n = "*execute this in parallel with the following*"

...tructions to optimize exec.

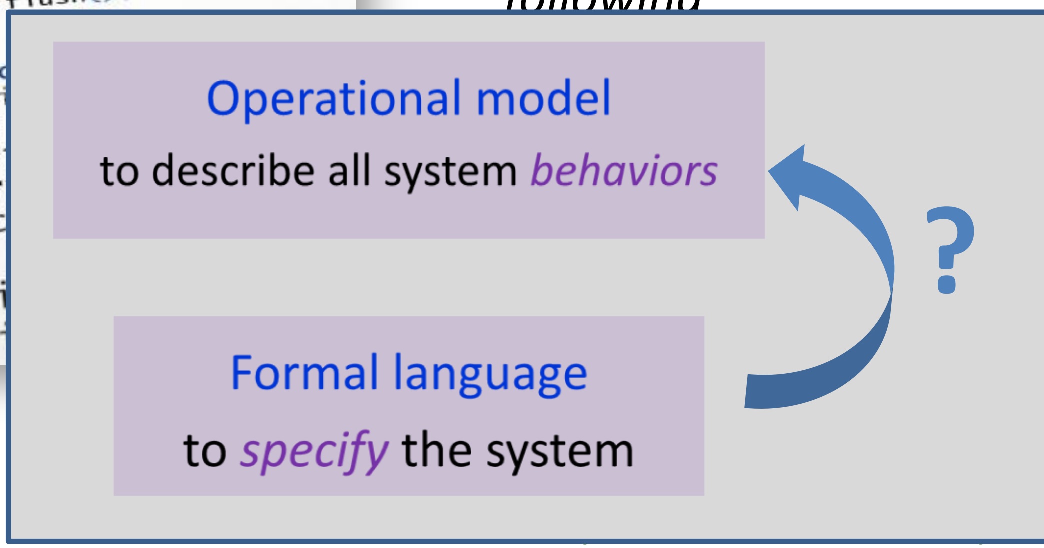
...*defines* a partial ordering

...*e happens-before relation*

... respect the HB-relation

...ulticore CPUs, GPUs)

...odel



- many approaches, research issue
- no causal discovery
- debate about how to define a precedence relation

Causality vs Dependency

- causal relation < encodes ***any form of dependency*** (temporal, spatial, causal,…)
  - well suited for the study of independent (i.e. concurrent) actions
  - this might be ok: it is simple and effective in many cases
    - e.g. independent sets of instructions can be *scheduled at the same time* over different CPU cores
  - in biological systems *causality* $\neq$ *necessary conditions* (knock-out causality) hence a formal treatment of causality like that in PESs, must be specialized

The meaning of causality ***in philosophy***

- puzzling about ***the nature of the connection*** from the cause and the effect
  - does the event A cause the event B in the sense of ***producing*** it?
  - what is the causal ***mechanism*** that is responsible for a phenomenon?

The meaning of causality ***in philosophy***

- puzzling about ***the nature of the connection*** from the cause and the effect
  - does the event A cause the event B in the sense of ***producing*** it?
  - what is the causal ***mechanism*** that is responsible for a phenomenon?
    - e.g. in **biology**: a virus *produces* flu, and we are interested in understanding the *mechanism* of spread of an infection
    - in **physical** processes: production is identified in the exchange of conserved quantities [Salmon-Dowe]
    - in **social** contexts: production is identified in terms of interaction between individuals, role of norms and values [Hedstrom-Ylikoski]

The meaning of causality ***in philosophy***

- puzzling about ***the nature of the connection*** from the cause and the effect

most philosophers agree that, in various scientific contexts,
causality involves
a ‘*dependence*’ component **AND** a ‘*productive*’ component

The meaning of causality ***in philosophy***

- puzzling about ***the nature of the connection*** from the cause and the effect

most philosophers agree that, in various scientific contexts, causality involves
a '*dependence*' component **AND** a '*productive*' component

In computer science in many cases:

causality / dependence / precedence / necessary condition
seem to be used as synonyms

deliberately ?

Reasoning above systems

Analysis techniques

to reason/prove system *properties*

Operational model

to describe all system *behaviors*

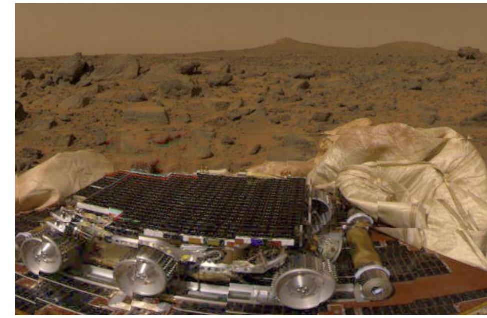
Formal language

to *specify* the system

a **property** is a proposition that holds true **in any execution** of the system

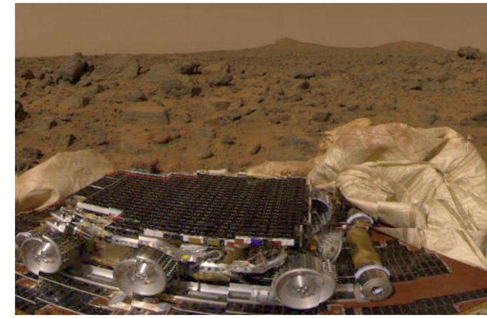
- concurrent systems allows many different executions:
  - A | B can be scheduled in any order
- real models are huge, possibly unbound
  - models are only partially built, possibly on-the-need
  - an exhaustive look is unfeasible.

counterfactuals at work



- 4th July 1997 Mars Pathfinder landed on Mars. The Sojourner rover started gathering and transmitting data back to Earth
- After few days the spacecraft began experiencing system resets
- NASA engineers spent hours running the replicated system in their lab attempting to replicate the precise conditions under which they believed that the reset occurred.
- When they finally reproduced a system reset on the replica, the analysis of the computation trace revealed a well-known concurrency bug, i.e. priority inversion.

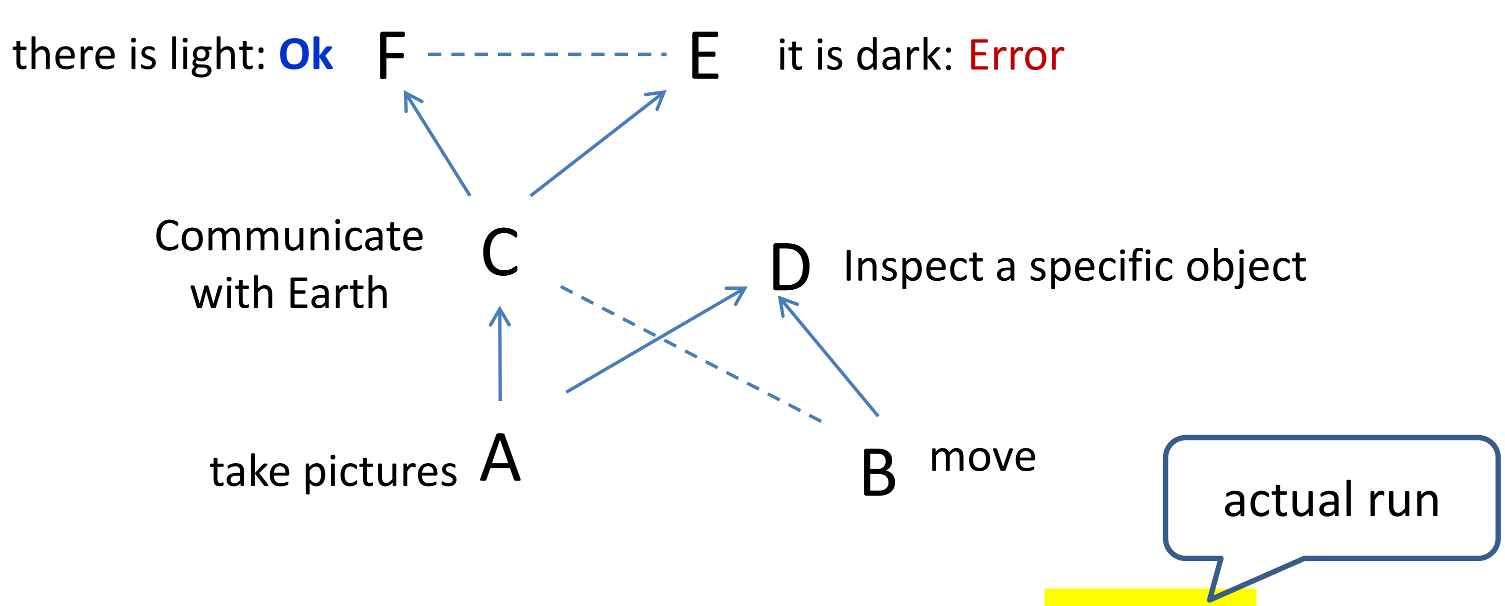
counterfactuals at work



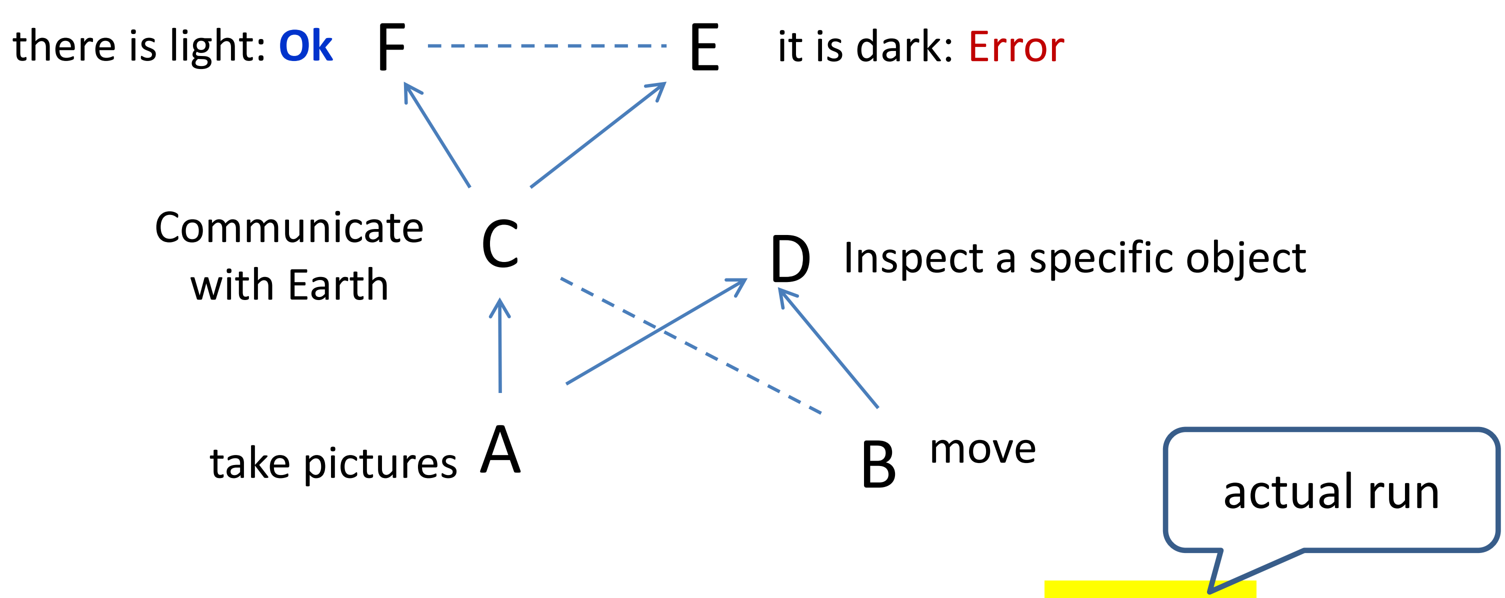
- 4th July 199… The Sojourner rover started gat… Earth
- After few d… …ng system resets
- NASA engineers spent hours running the replicated system in their lab attempting to replicate the precise conditions under which they believed that the reset occurred.
- When they finally reproduced a system reset on the replica, the analysis of the computation trace revealed a well-known concurrency bug, i.e. priority inversion.

they looked at (the huge) system model until they found a behavior ending up in the error state

Rephrase in terms of ***counterfactual reasoning*** on top of the (concurrent) operational model



- possible runs: A .B . D or B . A . D or (A|B).D or A . C . F or **A . C . E**
- The (huge) model had not been entirely built hence the error state went unnoticed

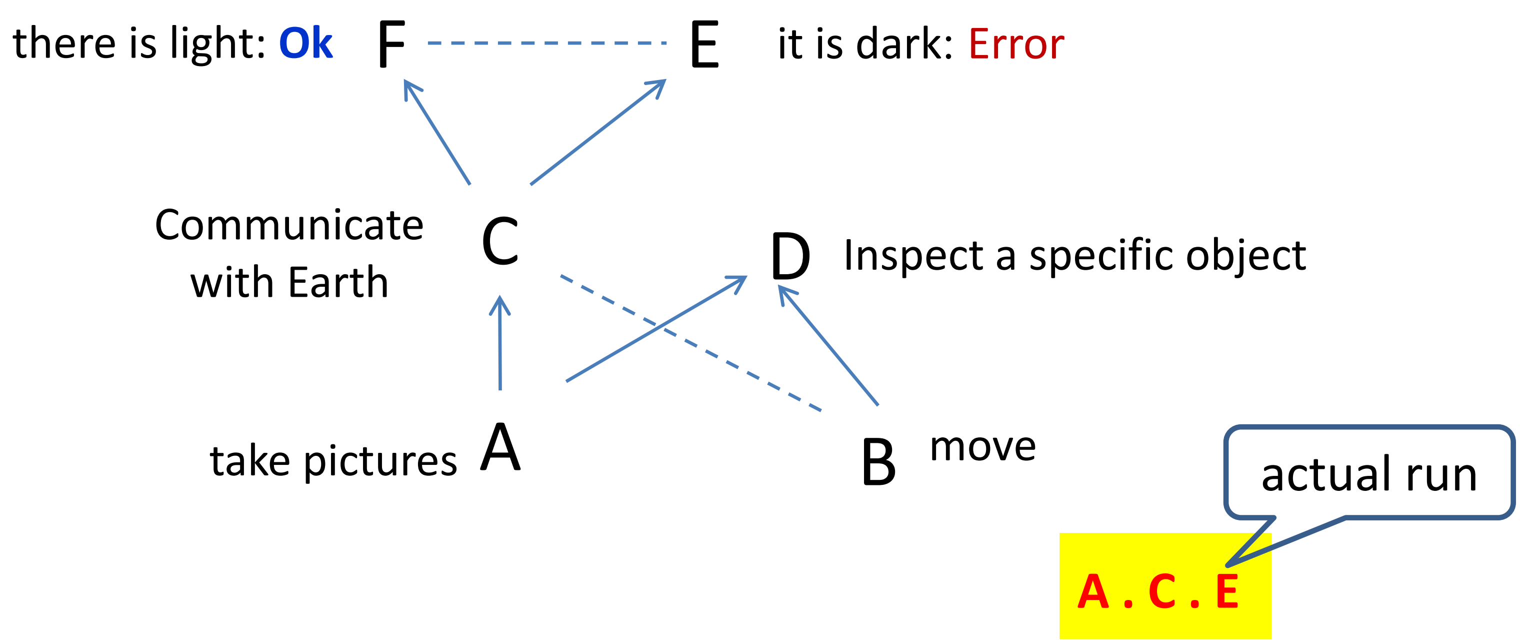


- possible runs: A .B . D or B . A . D or (A|B).D or A . C . F or **A . C . E**
- The (huge) model had not been entirely built hence the error state went unnoticed

we validate it using **N.Resher's theory**

Error explanation:

Since it was dark, **if the first Rover's action had been B**, it would not have entered the error state

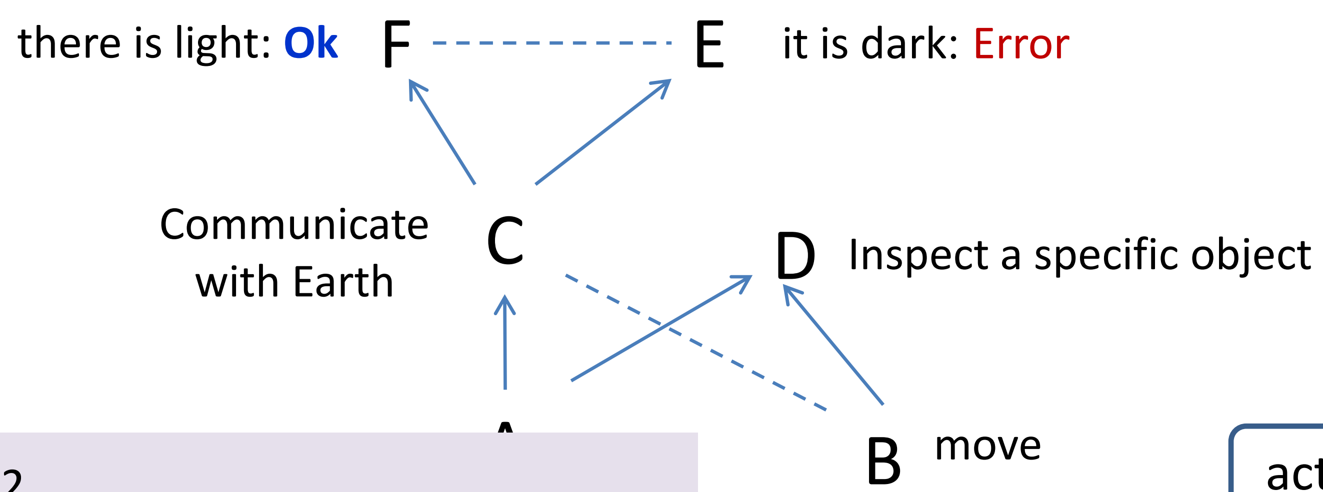


*Since it was dark, **if the first Rover's action had been B,** it would not have entered the error state*

we validate it using **N.Resher's theory**

List of the salient beliefs:

1. it was dark — fact
2. the Rover did not perform B as its first action — fact
3. the Rover performed C — fact
4. the Rover ended up in E — fact
5. The execution of B prevents the execution of E — law
6. If E is executed, then it is dark and C has been previously executed — law



assume not 2
then 4 and 5 are incompatible
reject 4: 5 is a law hence it has priority over 4

actual run

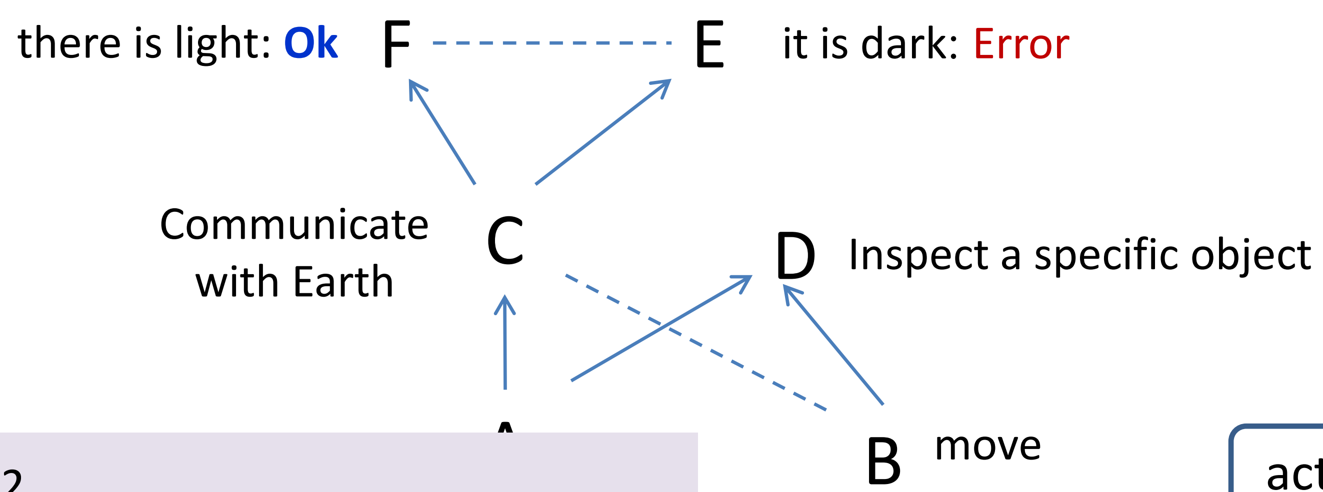
A . C . E

*Since it was dark, **if the first Rover's action had been B,** it would not have entered the error state*

we validate it using **N.Resher's theory**

List of the salient beliefs:

1.	it was dark	fact
2.	~~the Rover did not perform B as its first action~~ B is the first action	fact
3.	the Rover performed C	fact
4.	the Rover ended up in E	fact
5.	The execution of B prevents the execution of E	law
6.	If E is executed, then it is dark and C has been previously executed	law



assume not 2
then 4 and 5 are incompatible
reject 4: 5 is a law hence it has priority over 4

actual run

A . C . E

*Since it was dark, **if the first Rover's action had been B,** it would not have entered the error state*

we validate it using **N.Resher's theory**

List of the salient beliefs:

1. it was dark — fact
2. ~~the Rover did not perform B as its first action~~ B is the first action — fact
3. the Rover performed C — fact
4. ~~the Rover ended up in E~~ The rover does not end up in E — fact
5. The execution of B prevents the execution of E — law
6. If E is executed, then it is dark and C has been previously executed — law

Counterfactual reasoning

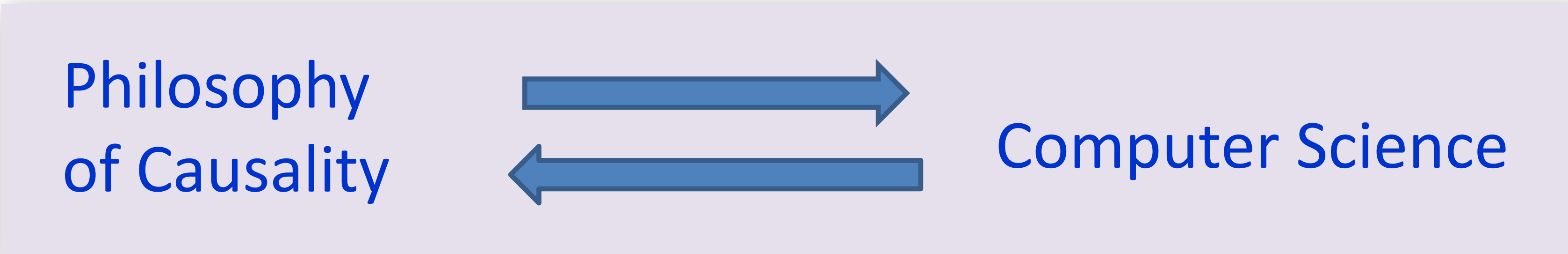
- Resher's account **well fits model-based trace analysis**
  - the different priority levels (*Meaning, Existence, Lawfulness, Fact*) boil down to the distinction between *facts* ("event E occurred"), and *laws* ("A and B are independent")
  - with such a clear distinction we can always decide the priority of beliefs, while in Lewis' theory the similarity between possible worlds is an open problem

Counterfactual reasoning

- Resher's account **well fits model-based trace analysis**
  - the different priority levels (*Meaning, Existence, Lawfulness, Fact*) boil down to the distinction between *facts* ("event E occurred"), and *laws* ("A and B are independent")
  - with such a clear distinction we can always decide the priority of beliefs, while in Lewis' theory the similarity between possible worlds is an open problem

- the model can be used to **refute** a counterfactual by showing a possible execution violating the c.
  - *If the first action had been A, it would have ended up in error state*
  - Show an allowed behavior where the counterfactual is false: A – B – D
  - Resher doesn't refer to c. refutation, but only to proving c. negation (deinal) *If the first action had been A, it would NOT have ended up in error state*

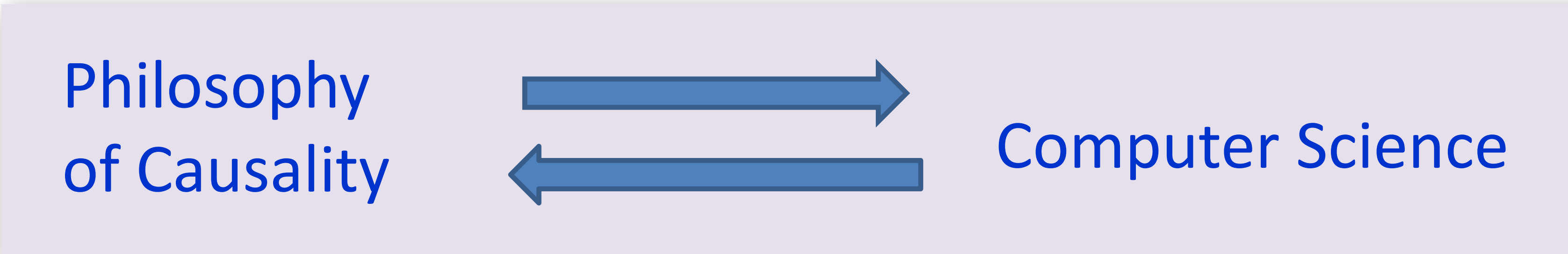
Conclusions



- The formalization of concurrent systems is an interesting area where to investigate the meaning and the use of causal concepts

- Causal talking is used in many other approaches to concurrent systems , each one with its peculiarities

Conclusions



- The formalization of concurrent systems is an interesting area where to investigate the meaning and the use of causal concepts

- Causal talking is used in many other approaches to concurrent systems , each one with its peculiarities

- We don't aim to be general, but
  - to point out how tricky and subtle is causal talking, even in Computer Science
  - to build a bridge with the philosophy of causality developed in other scientific contexts

THE END