Causality in Concurrent Systems

F. Russo
Vrije Universiteit Brussel
Belgium

S. Crafa
Università di Padova
Italy

HaPoC 31 October 2013, Paris
software, hardware or even physical systems where sets of activities \textit{run in parallel} with possible occasional \textit{interactions}
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

• Java programming language, ...
• DSL for concurrent hardware, system biology, ...
• process algebras

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

Operational model
to describe all system *behaviors*

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

Formal language
to *specify* the system
Overview

• Concurrent Systems
  – How to deal with such 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

what causality means? w.r.t. traditional debates in philosophy of causality:
  production and mechanisms, independence, causation by omission

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*

Operational model
to describe all system *behaviors*

Formal language
to *specify* the system

• *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)
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*

Operational model
to describe all system *behaviors*

• *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)

**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

- 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

The presence of Train1 at C depends on its previous presence at B
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.
• **Labeled Prime Event Structure** \(<E,\prec,\#$,\lambda>\)
  
  – \(E\) is a set of events \(e\) (event=a step of computation)
  – \(\lambda(e)\) action associated to the occurrence of \(e\)
  
  – \(\prec\) a **partial order** representing the **causal relation** between events:
    • \(e_1 \prec e_3\)   \(e_1\) is a cause of \(e_3\)
  
  – \(\#\) irreflexive and symmetric relation called **conflict**:
    • \(e_3 \# e_4\)   two alternative behaviors
  
  – axiom: the conflict is hereditary: if \(e \prec e'\) and \(e \# e''\) then \(e' \# e''\)
The railway system as a PES

- **Labeled Prime Event Structure** (flow e.s., asymmetric conflict,...)
- Petri nets (multiple tokens, open nets, ...)
- generalized Labeled Transition Systems
- (unstable) configuration structures
- causal trees
- ...
The railway system as a PES

- Labeled Prime Event Structure (flow e.s., asymmetric conflict,...)
- Petri nets (multiple tokens, open nets, ...)
- Generalized Labeled Transition Systems
- (unstable) configuration structures
- Causal trees
- ...
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:**
  
  \[ e_1 \text{ causally depends on } e_2 \iff \text{it has been so defined} \]

• 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:**
  
  \[ e_1 \text{ causally depends on } e_2 \iff \text{it has been so defined} \]

- 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’

**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 \( n \)
  - \textit{ok} in sequential code
  - \textit{but}: instruction \( n = "execute \ this \ in \ parallel \ with \ the \ following" \)
  - \textit{but}: runtime reorders instructions to optimize exec.

- many approaches, research issue
- \textbf{no causal discovery}
- debate about \textbf{how to define a precedence relation}
The meaning of causality in PESs

- instruction \( n+1 \) causally depends on instruction \( n \)
  - \( \text{ok} \) in sequential code
  - \( \text{but} \): instruction \( n = \) "execute this in parallel with the following"
  - \( \text{but} \): 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 \( n \)
  - \( \text{ok} \) in sequential code
  - \( \text{but:} \) instruction \( n = \text{“execute this in parallel with the following”} \)
- the \( \text{(Java)} \) 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
- 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 ≠ 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 allow 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 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.

Rephrase in terms of counterfactual reasoning on top of the (concurrent) operational model.
The (huge) model had not been entirely built hence the error state went unnoticed.


- There is light: Ok
- It is dark: Error

- Communicate with Earth
- Take pictures
- Inspect a specific object
- Move

Actual run: A . C . E
• The (huge) model had not been entirely built hence the error state went unnoticed

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.

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

List of the salient beliefs:

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

we validate it using N. Resher’s theory
Since it was dark, if the first Rover’s action had been B, it would not have entered the error state.

List of the salient beliefs:

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