Spatio-temporal modelling and simulation with the FCPP Aggregate Programming framework

I - Introduction

Volker Stolz,[†] Ferruccio Damiani,[*] Giorgio Audrito[*]

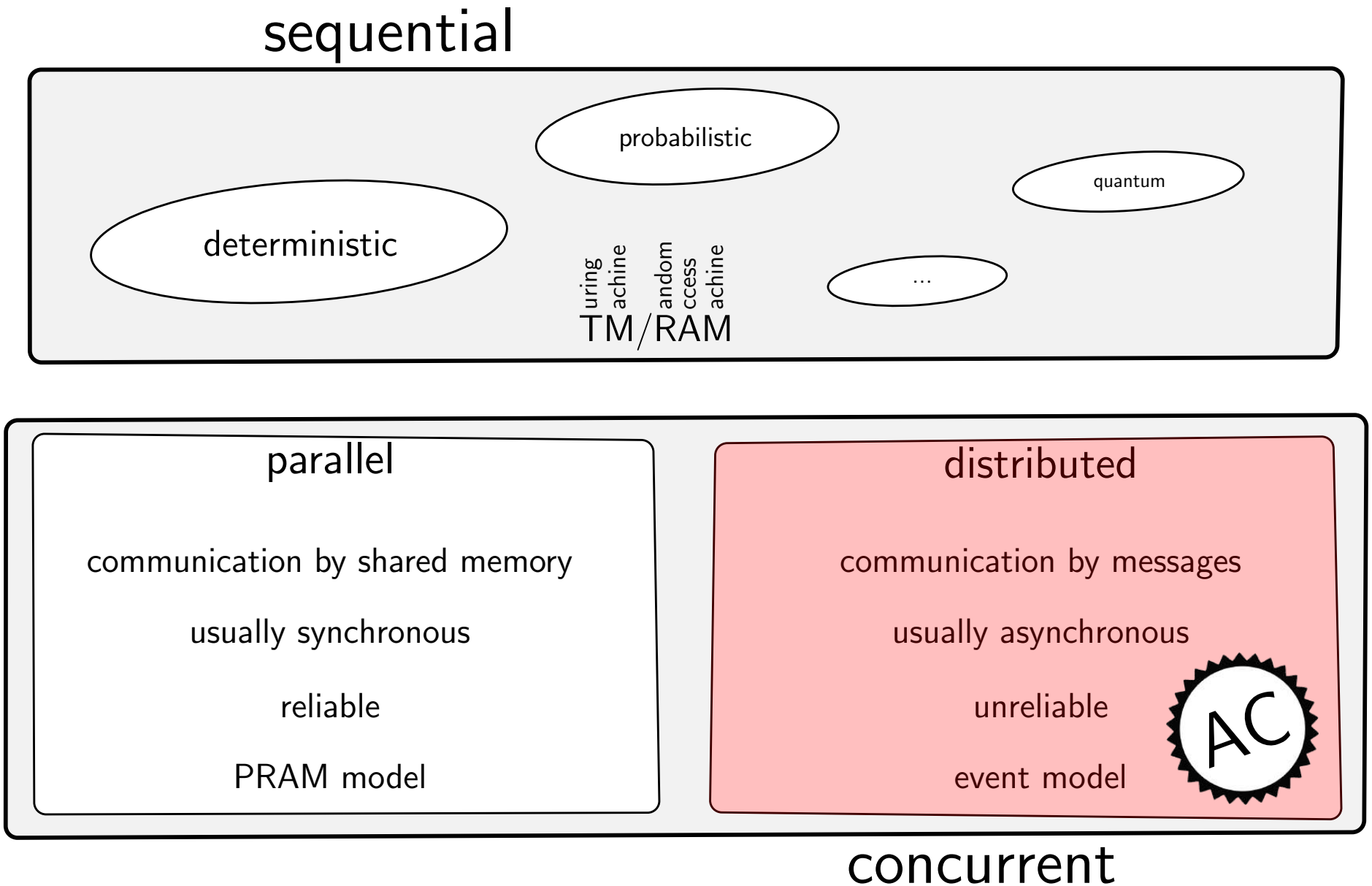
[*]University of Turin, Italy
[†]Western Norway University of Applied Sciences, Bergen

June 20, 2022

Outline

- What is Aggregate Computing about?
  - Which computing systems?
  - Which frameworks?
  - What kind of problems?
- Why care about Aggregate Computing?
- So, what Aggregate Computing is?
- How can we formalise it?

Which computing systems?



Which frameworks?

	centralised	**decentralised**
approach	few special central units responsible of computing and deciding, most units only sensing and actuating	all units are equal
collaboration	easy	hard
robustness	lower	higher
privacy	lower	higher

AC is a decentralised framework

Which frameworks?

	"query-based"	**"data-based"**
approach	computing required results whenever queries are issued	maintaining relevant data always updated and locally accessible
requirements	none	predictable queries
resource usage	on demand (best for fewer queries)	constant (best for frequent queries)
responsiveness	lower: requires routing	higher: data locally available
quality	higher: fresh results	lower: older results

AC is designed for "data-based" frameworks
(extensions handling query-based frameworks are possible)

What kind of problems?

(not only) spatial computing problems

position and time of devices are essential inputs of the computation

distance estimation, data summarisation (event detection),
selecting areas (network partitioning, channel establishment...),
inducing shapes (crowd dispersion, formation control...)... and others!

So, what is Aggregate Computing about?

- Which computing systems? $\longrightarrow$ distributed systems
- Which frameworks? $\longrightarrow$ decentralized, (mostly) data-based
- What kind of problems? $\longrightarrow$ born for (but not limited to) spatial computing

is a recent topic...

- cons: not standard, not well-known, small research group
- pros: plenty of things where research is needed

Outline

- What is Aggregate Computing about?
- Why care about Aggregate Computing?
  - Why care for distributed systems at all?
  - Why are distributed systems hard to deal with?
  - How Aggregate Computing can help?
- So, what Aggregate Computing is?
- How can we formalise it?

Why care for distributed systems at all?

- increasing availability of wearable / mobile / embedded / flying devices
- increasing availability of heterogeneous wireless connectivity
- increasing availability of computational resources (device/edge/cloud)
- increasing production/analysis of data, everywhere, anytime




Why are distributed systems hard to deal with?

diverse heterogeneous entities

- different computing power
- sensing and actuation capabilities

we need...

- device abstraction
- multi-platform frameworks
- not too bad so far...

Why are distributed systems hard to deal with?

data security and privacy

- localised data may be highly personal
- cryptography has got it covered, but hacker attacks do happen
- can we really trust big companies having our data?

in a decentralised system, we can. . .

- keep our data locally
- where communication is needed, share locally aggregated neighbour data

Why are distributed systems hard to deal with?

predictable code re-use

- can break down complexity of problems
- needs to avoid unwanted interferences

in a sequential system...

- machine code is hard to re-use
- abstractions: objects, mixins...
- all boil down to functions and composition

in a distributed system...

- need to match messages
- not good with process calculi
- ...how to handle multiple instances of same algorithm?

Why are distributed systems hard to deal with?

dynamic goals and environment

- low-end devices may be unreliable...
- as low-power wireless communications between them
- environmental factors may affect the computation

in a centralised system...

- changes in edge devices OK
- changes in central units BAD
- connection loss BAD

in a decentralised system...

- any change OK but HARD
- need re-usable algorithms...
- with precise adaptivity guarantees

Why are distributed systems hard to deal with?

collaboration vs selfishness

- many devices and goals to be achieved
- competitive behaviour may prevent to achieve them
- much easier in centralised systems

in a decentralised system...

- we want to achieve a given global behaviour,
- by programming local interactions
- seek for an automatic global to local translation

How Aggregate Computing can help?

challenges are...

- diverse heterogeneous entities
- data security and privacy
- predictable code re-use
- dynamic goals and environment
- collaboration vs selfishness

aggregate computing provides...

- a multi-platform framework
- ...which is decentralized
- composable abstraction of behaviour
- ...and guarantees
- with global-to-local translation

Outline

- What is Aggregate Computing about?
- Why care about Aggregate Computing?
- So, what Aggregate Computing is?
  - A computation model for proximity-based networks
  - What it provides?
  - Properties
  - How developing works?
- How can we formalise it?

A computation model for proximity-based networks

Network of (possibly mobile) devices

- dynamic topology — induced by (physical or logical) proximity of devices
- each device can receive/send message to/from devices in the neighbourhood

where each device executes a computation/coordination service S

To execute S, each device (asynchronously periodically) fires:

1. collects local (sensor, DB,...) data and received messages
2. executes a program — the same for all devices[1]
3. sends messages (and performs some actuation)

[1] Supports integrated development of every part of a distribute application – cf. *macroprogramming* and *multi-tier programming*:
• R. Newton and M.Welsh (2004). *Region streams: Functional macroprogramming for sensor networks*. In: 1st Workshop on Data Management for Sensor Networks (MSN 2004), pp. 78–87. https://doi.org/10.1145/1052199.1052213
• P. Weisenburger, M. Köhler, G. Salvaneschi (2018). *Distributed system development with Scalaloci.* In: ACM Programming Languages, 2 (OOPSLA):129:1–129:30, 2018. https://doi.org/10.1145/3276499

What it provides?

1. Formal language and models for studying algorithms and properties

$$\texttt{e} ::= \texttt{x} \mid \phi \mid \texttt{c}(\overline{\texttt{e}}) \mid \texttt{b} \mid \texttt{d} \mid (\overline{\texttt{x}})\texttt{=>e} \mid \texttt{e}(\overline{\texttt{e}}) \mid \texttt{nbr}\{\texttt{e}\} \mid \texttt{old}(\texttt{e})\{\texttt{x=>e}\} \ldots$$

2. Language implementations (Proto, Protelis, ScaFi, FCPP)

```
DEF() double abf(ARGS, bool source) { CODE
    return nbr(CALL, INF, [&] (field<double> d) {
        double v = source ? 0.0 : INF;
        return min_hood(CALL, d + node.nbr_dist(), v);
    });
}
```

3. Network simulator (Alchemist, FCPP) or real deployment (FCPP)

Properties

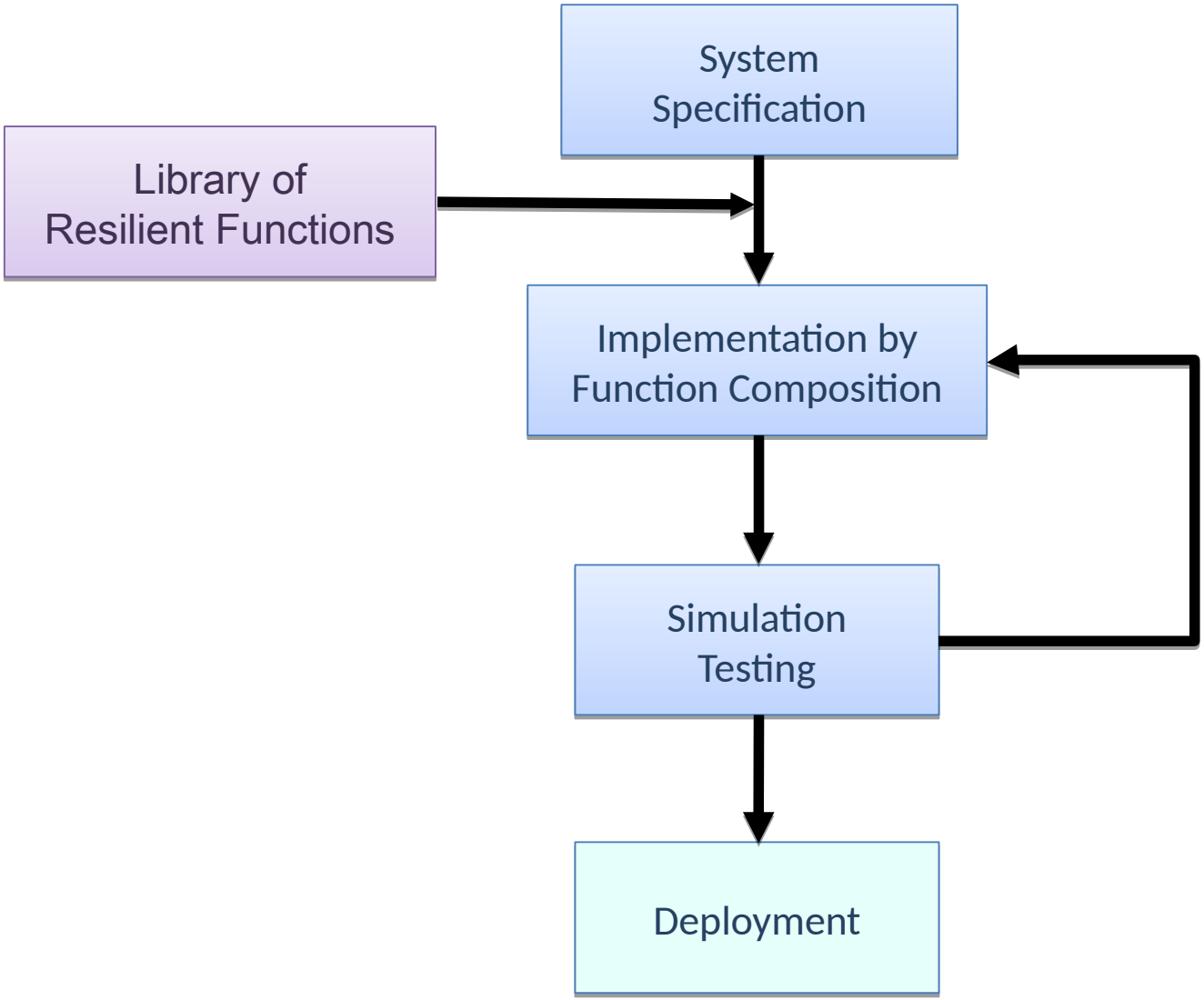
- Self-stabilisation: computation resilient to changes in devices state and network topology (behaviour is guaranteed to eventually attain a correct and stable final state despite any transient perturbation in state or topology)[1]
- Eventual consistency: computation independent from device number and density (behaviour converges to a final state that approximates a predictable limit, based on the continuous environment, as the density and speed of devices increases)[2]
- Certified error bounds: how computation reacts to changes (linking the quality of the services to the amount of computing resources dedicated)[3]
- ...

[1] M. Viroli, G. Audrito, J. Beal, F. Damiani, D. Pianini (2018). *Engineering resilient collective adaptive systems by self-stabilisation.* ACM Transactions on Modeling and Computer Simulation, 28(2), 2018. https://doi.org/10.1145/3177774

[2] J. Beal, M. Viroli, D. Pianini, F. Damiani (2017). *Self-adaptation to device distribution in the internet of things.* ACM Transactions on Autonomous and Adaptive Systems 12(3), 2017. https://doi.org/10.1145/3105758

[3] G. Audrito, F. Damiani, M. Viroli, E. Bini (2018). *Distributed Real-Time Shortest-Paths Computations with the Field Calculus.* In: IEEE 39th Real-Time Systems Symposium (RTSS 2018). https://doi.org/10.1109/RTSS.2018.00013

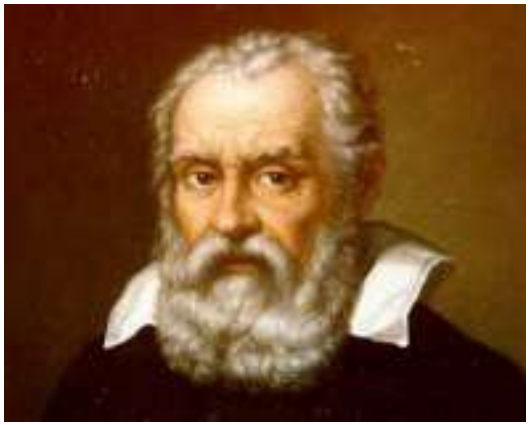
How developing works?



Outline

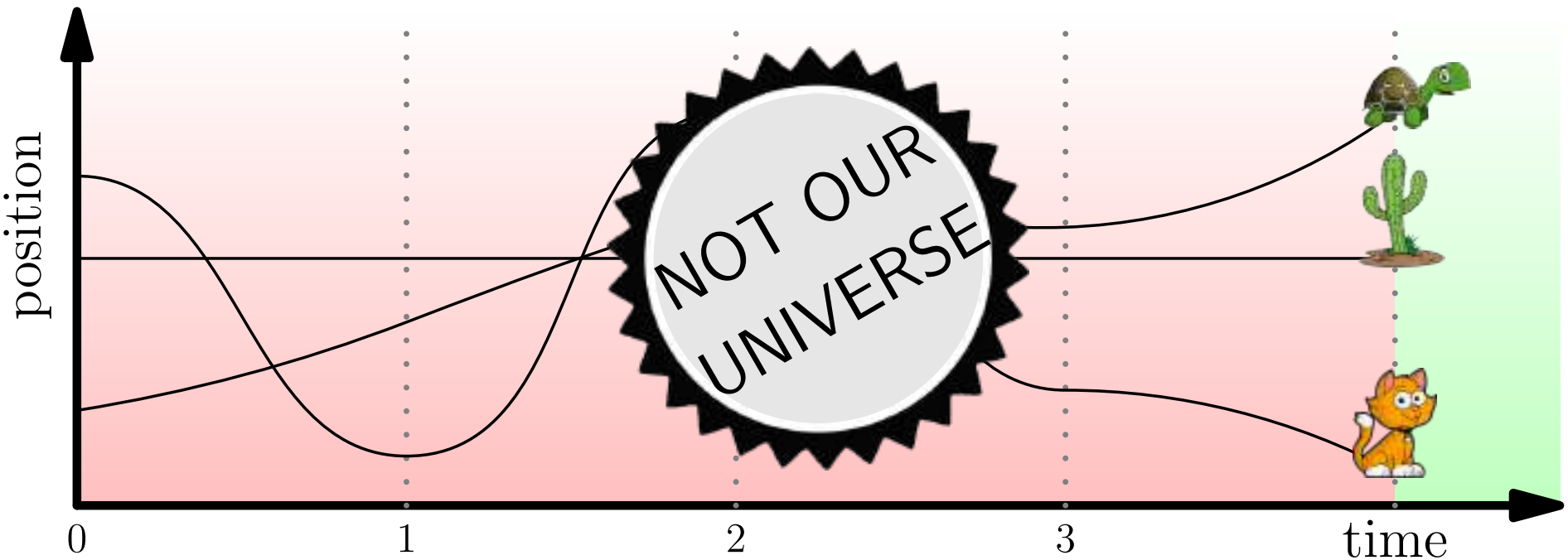
- What is Aggregate Computing about?
- Why care about Aggregate Computing?
- So, what Aggregate Computing is?
- How can we formalise it?
  - What physics can say
  - Event structures
  - Space-time values
  - Space-time functions

What physics can say



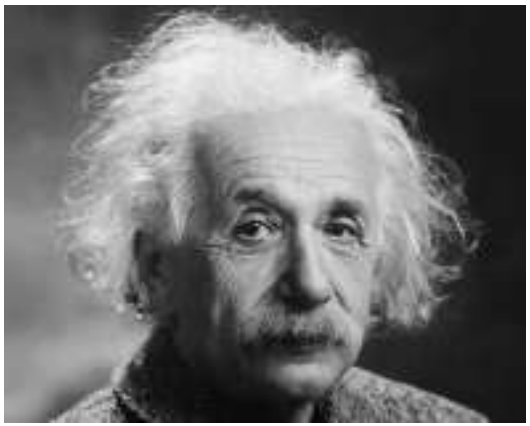
Galileo, help!

- events are pairs position, time $\langle \vec{x}, t \rangle$
- position is relative, but time is universal
- past, present and future are linear



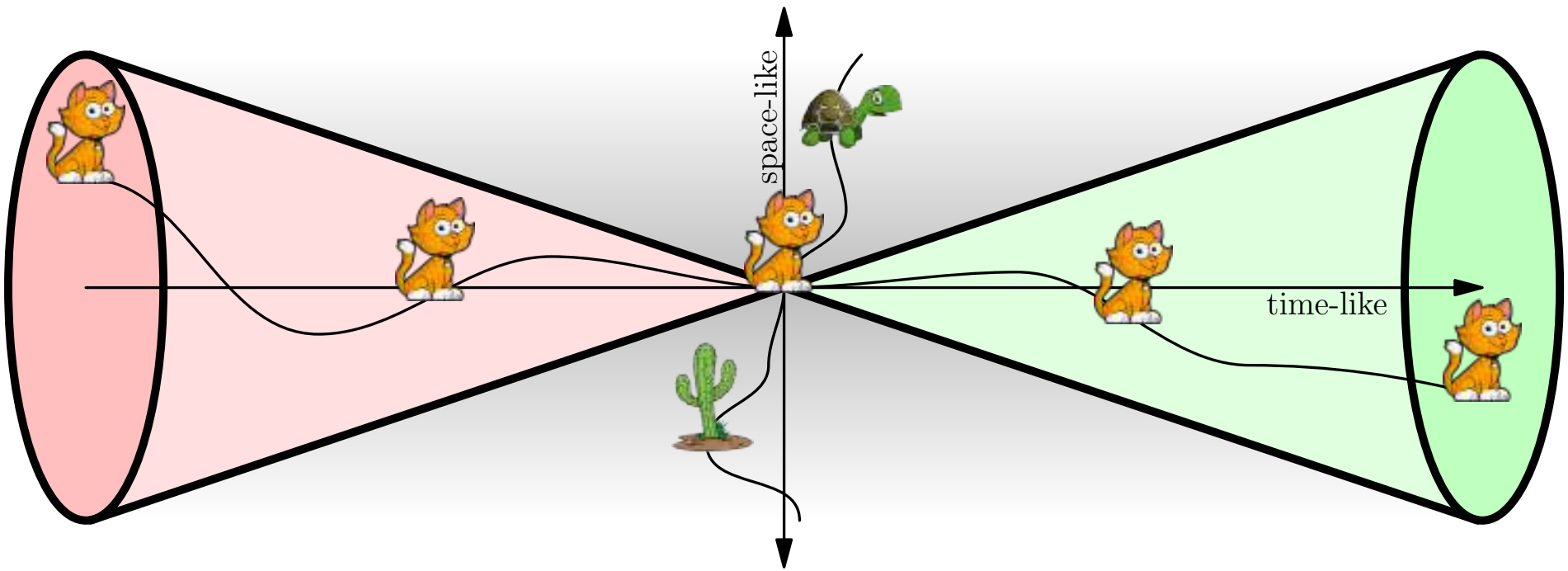
sounds like a sequential/parallel system...

What physics can say



Einstein, help!

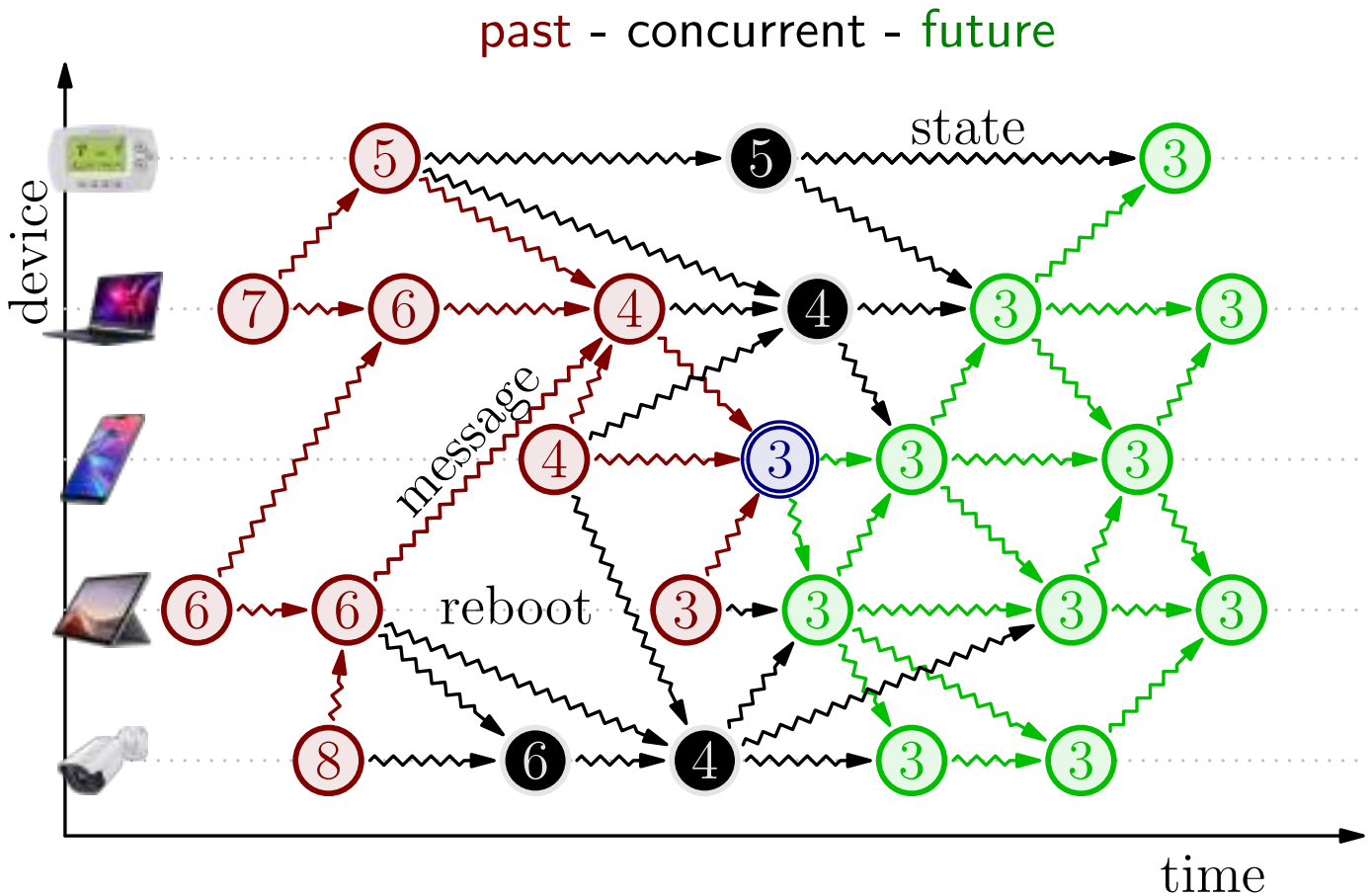
- position and time are both relative
- no clock, only relation between events
- some events are just not comparable



sounds weird just like a distributed system!

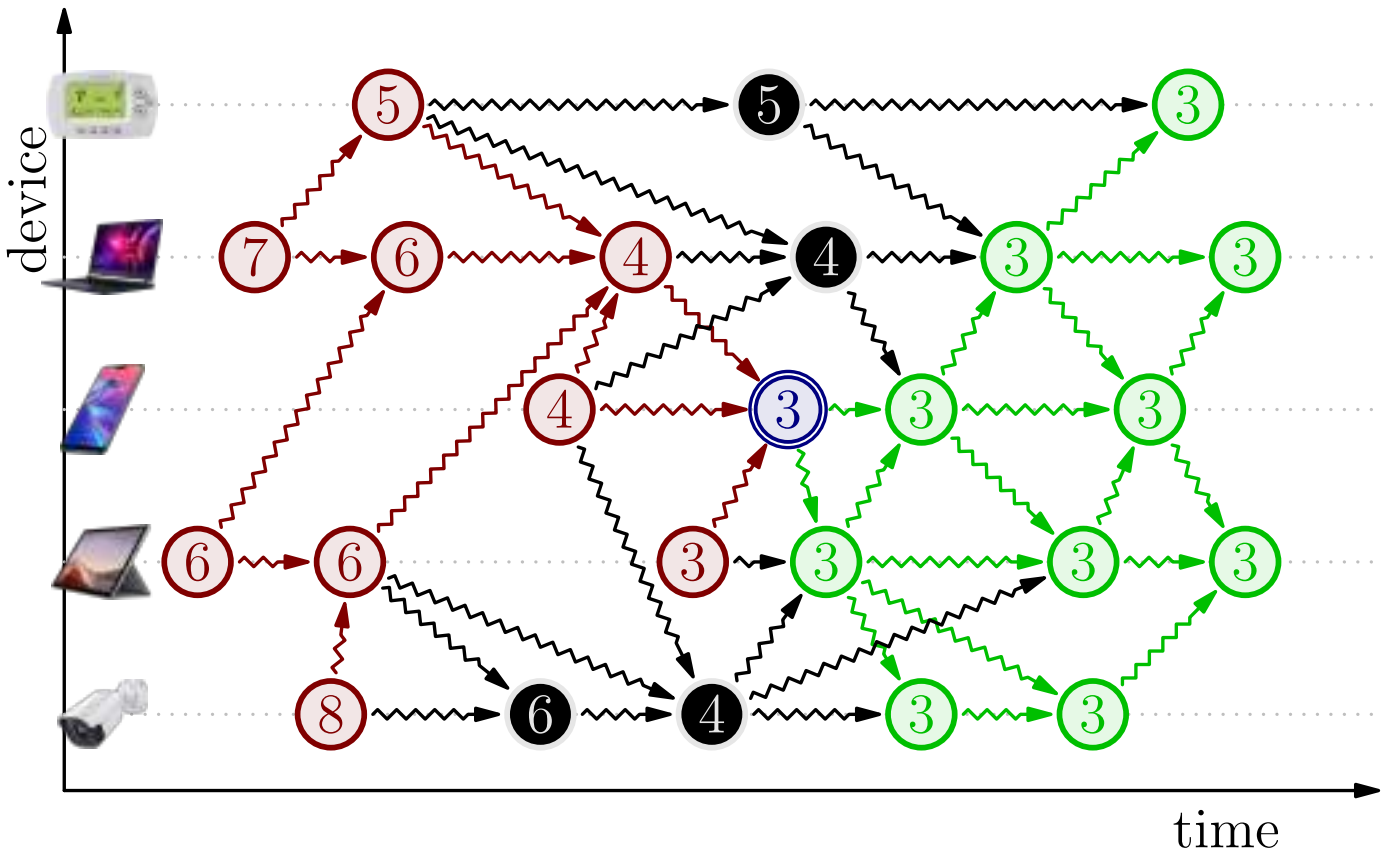
Event structures

- a set of events E
- a DAG of messages $\rightsquigarrow$
- a causality partial order $<$ (transitive closure of $\rightsquigarrow$)



Space-time values

- maps from events in event structures to values $\Phi : E \to X$

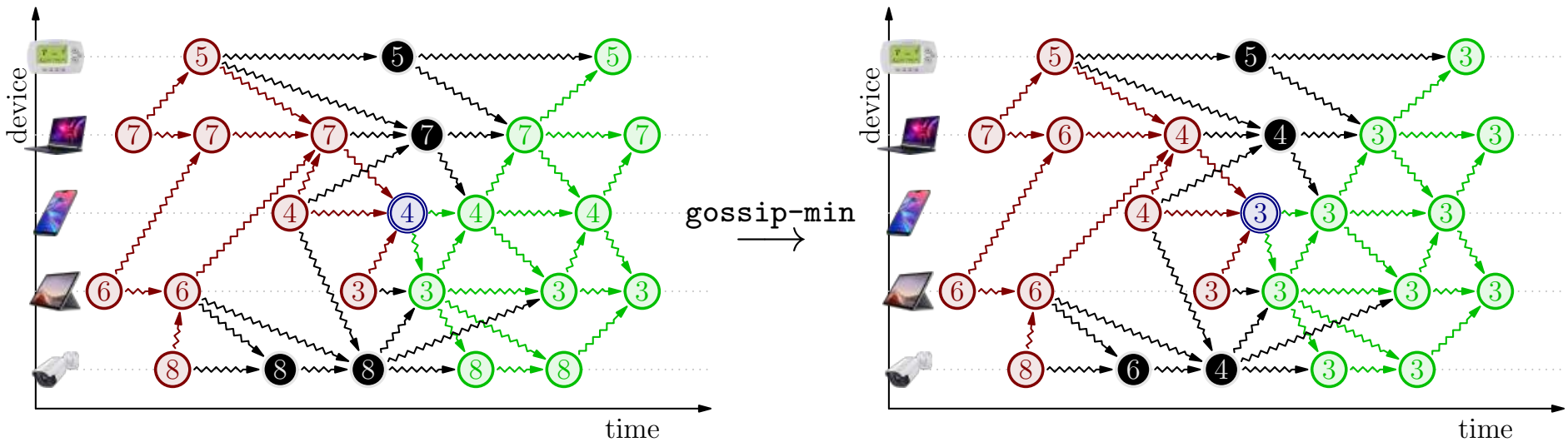


↑ space-time value resulting from a "gossip-min"

every device gossips the minimum of an input value and neighbours' gossips

Space-time functions

- maps from space-time values to space-time values $f(\Phi_{\text{in}}, \ldots) = \Phi_{\text{out}}$



Φ_{in} $\qquad$ $\texttt{gossip-min}(\Phi_{\text{in}}) = \Phi_{\text{out}}$

- they're just plain old functions (maybe with weird domain)
- we can compose them as usual $f(g(\Phi_1), h(\Phi_2), \Phi_3)$