

Semantic Models and Formal Analysis of Distributed/Global Algorithms

EPFL
& SNSF
& Hasler foundation

EU FET GC
PEPITO

Uwe Nestmann

Concurrency Theory / Applied Formal Semantics

*Thrust in Reliable Software Research (TRESOR)
School of Computer & Communication Sciences (I&C)
Swiss Federal Institute of Technology Lausanne (EPFL)*

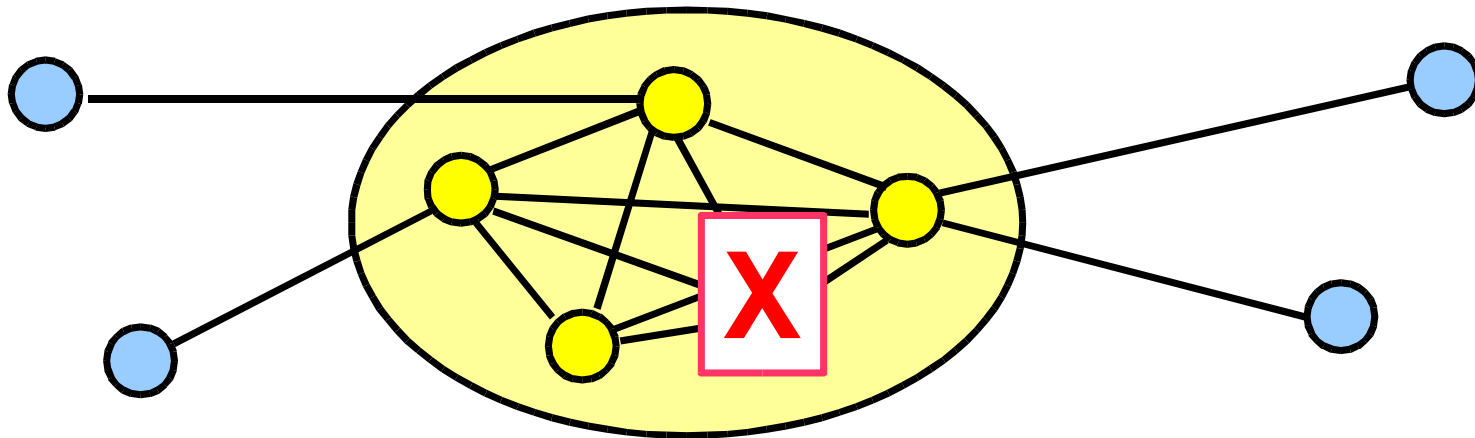
Extracting Guarantees From Chaos

John Kubiatoicz
Oceanstore @ Berkeley
CACM 46(2), Feb 2003.

Distributed Alg. Case Study: Group Communication

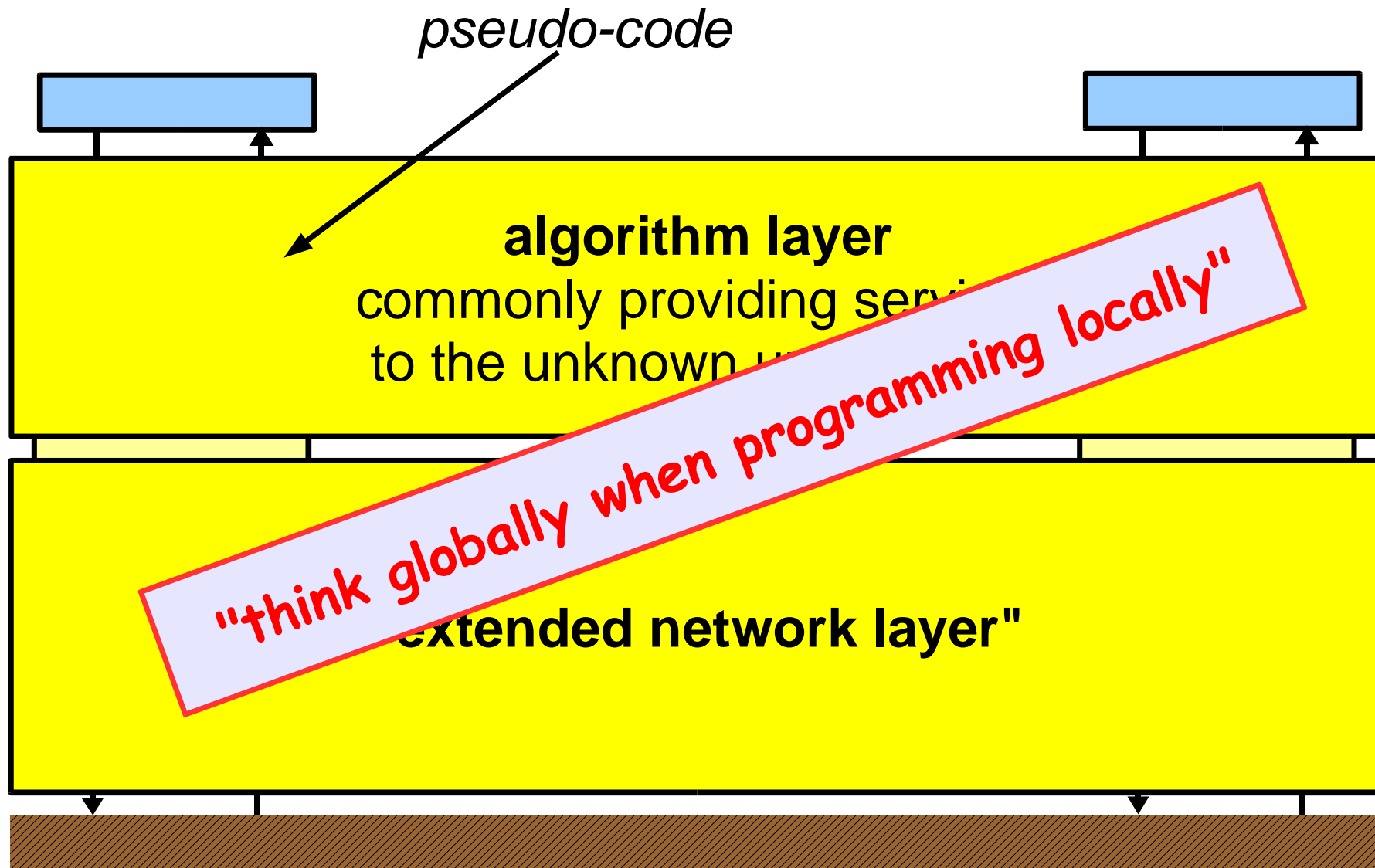
programming abstractions

that allow a *distributed group* of processes to together provide a *reliable common service* in spite of the possibility of *failures within the group*



- *membership* protocols
- *broadcast* protocols (multi, reliable, atomic, ...)
- *agreement* protocols (consensus, leader election, ...)

Layered Design of GC Stacks



Correctness of Consensus

Consensus is about the guaranteed possibility to reach an **agreement** among all processes on one of the values that is proposed by either of them.

Termination:

If

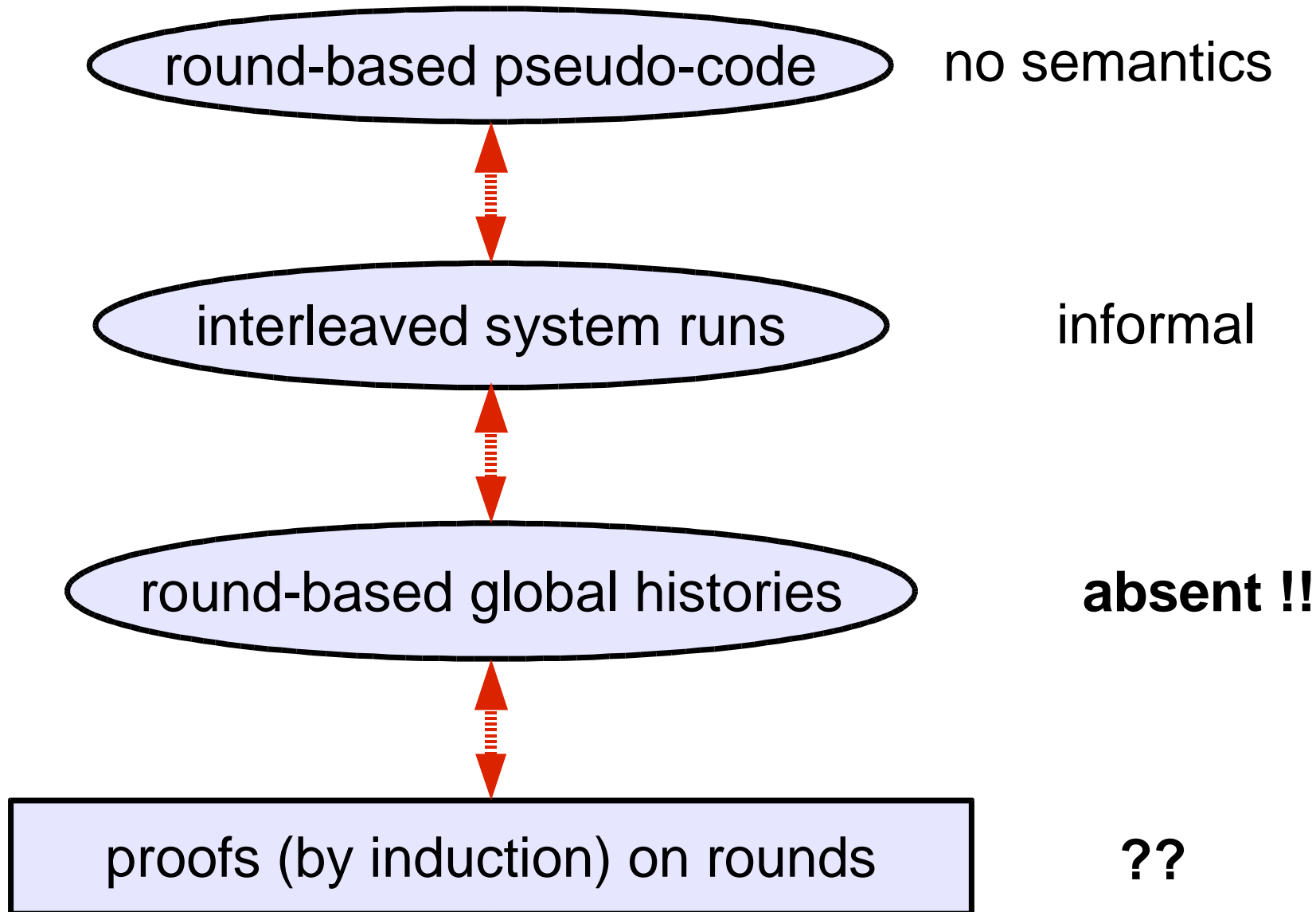
"in any run, there is a time t such that some correct process (i.e., one that will survive in that run) will afterwards never again be suspected"

then

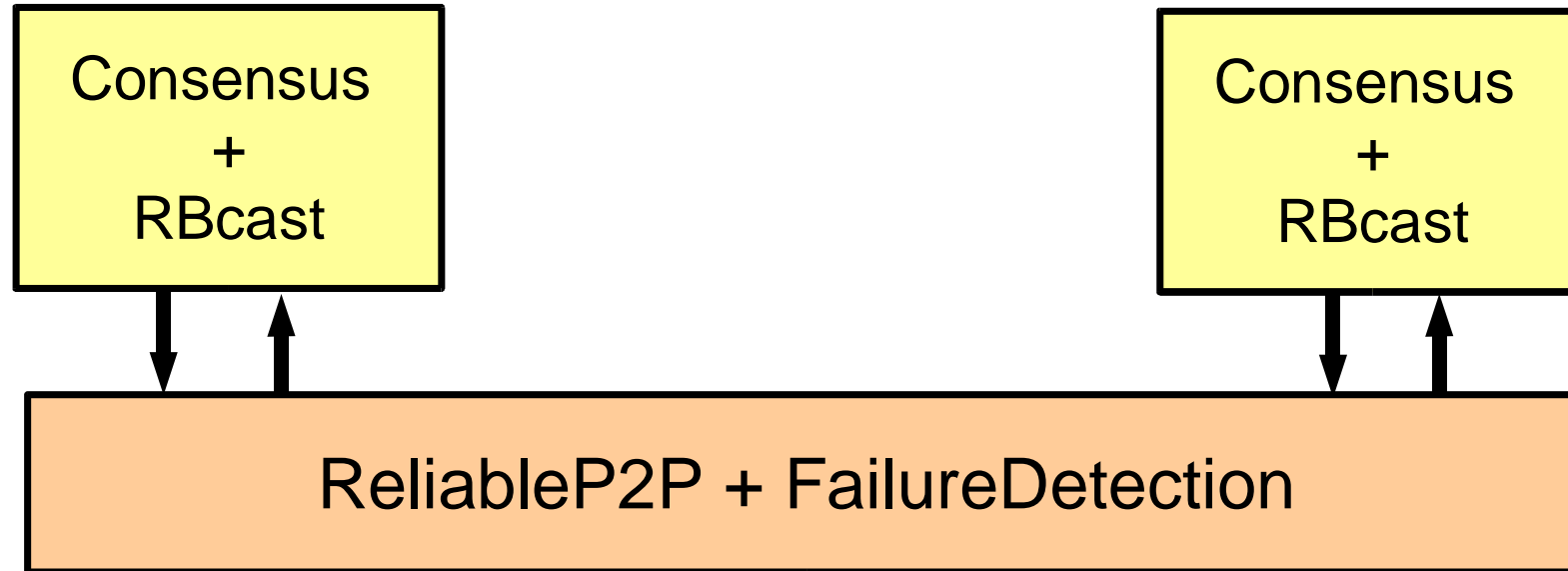
"in any run, every correct process will eventually decide for some value".

Agreement: ...

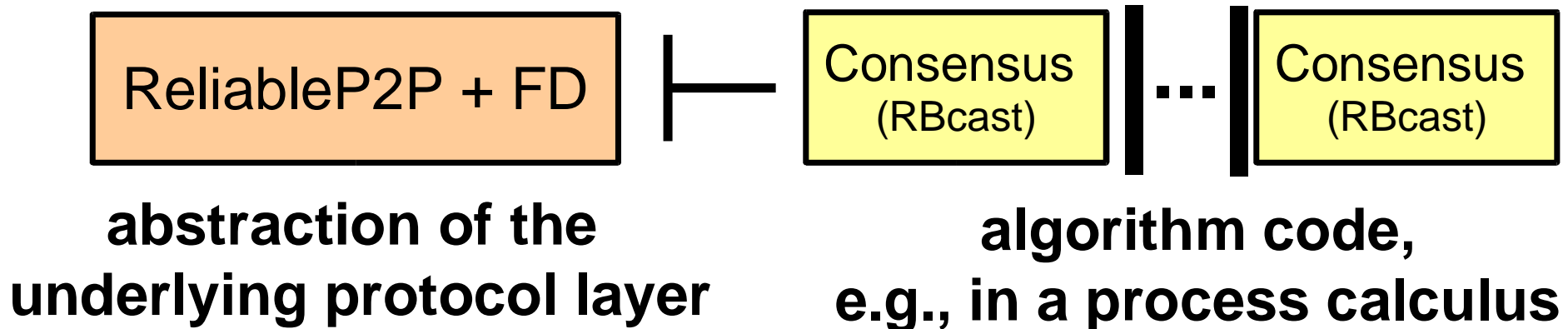
Usually ...



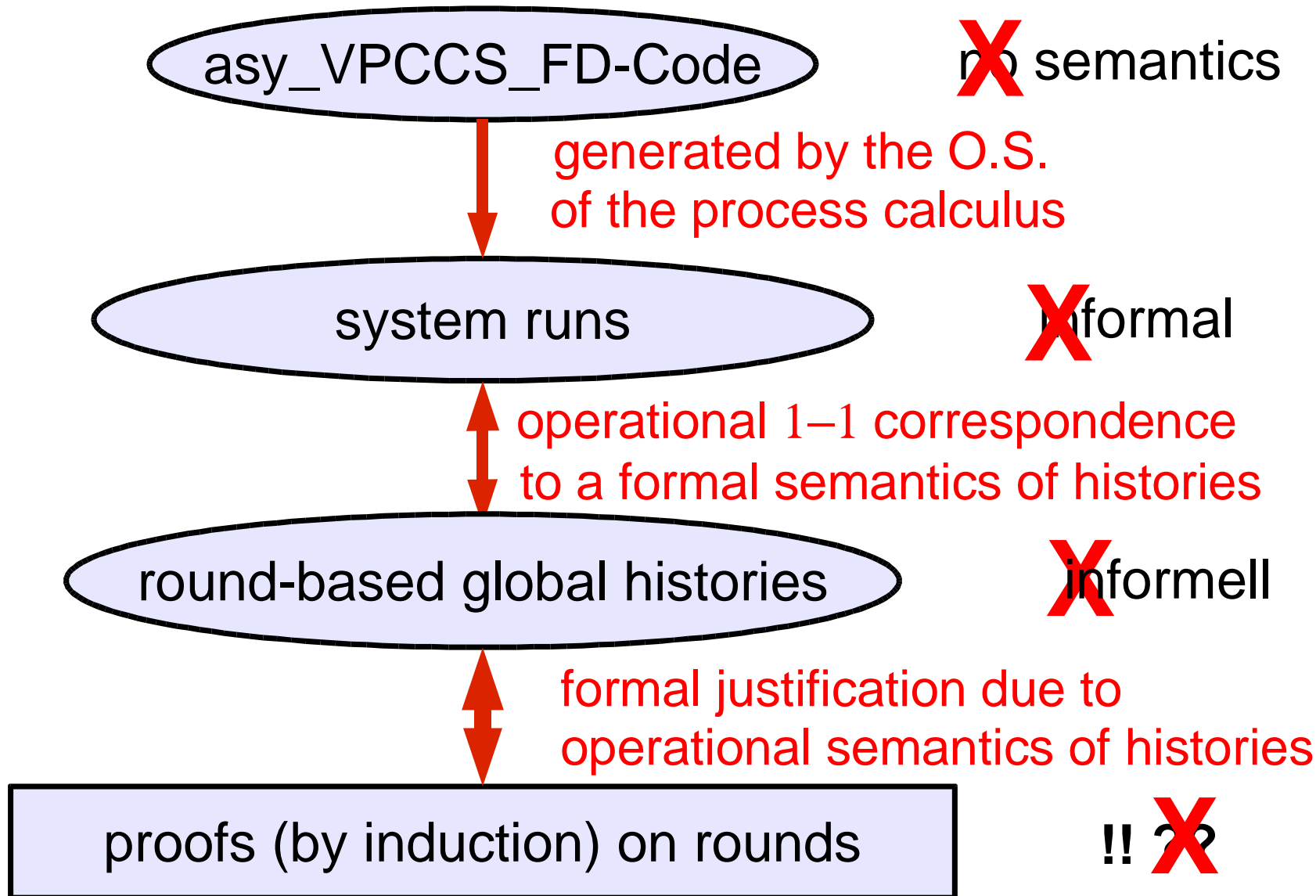
Layered Verification



is conveniently modeled as:



Last Year (CONCUR)



Now/Soon !

no explicit code

rule-based global
stateful description of the
dynamic behavior of the algorithm
and its possible histories

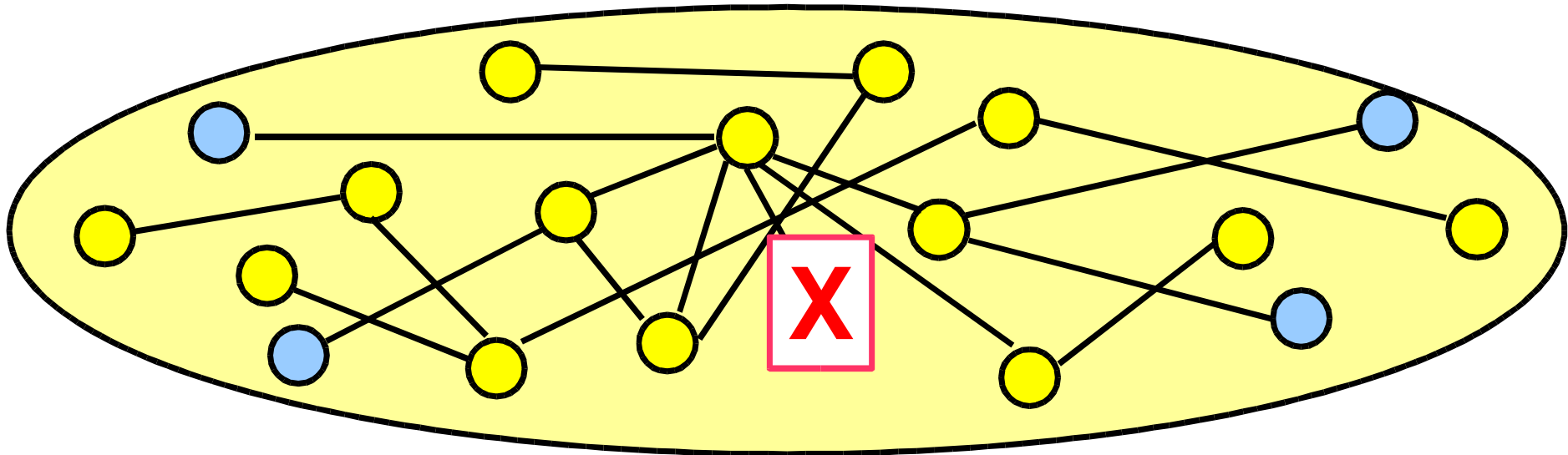


formal justification by definition

proofs (by induction) on rounds

Global Alg. Case Study: DHTs on P2P-Networks

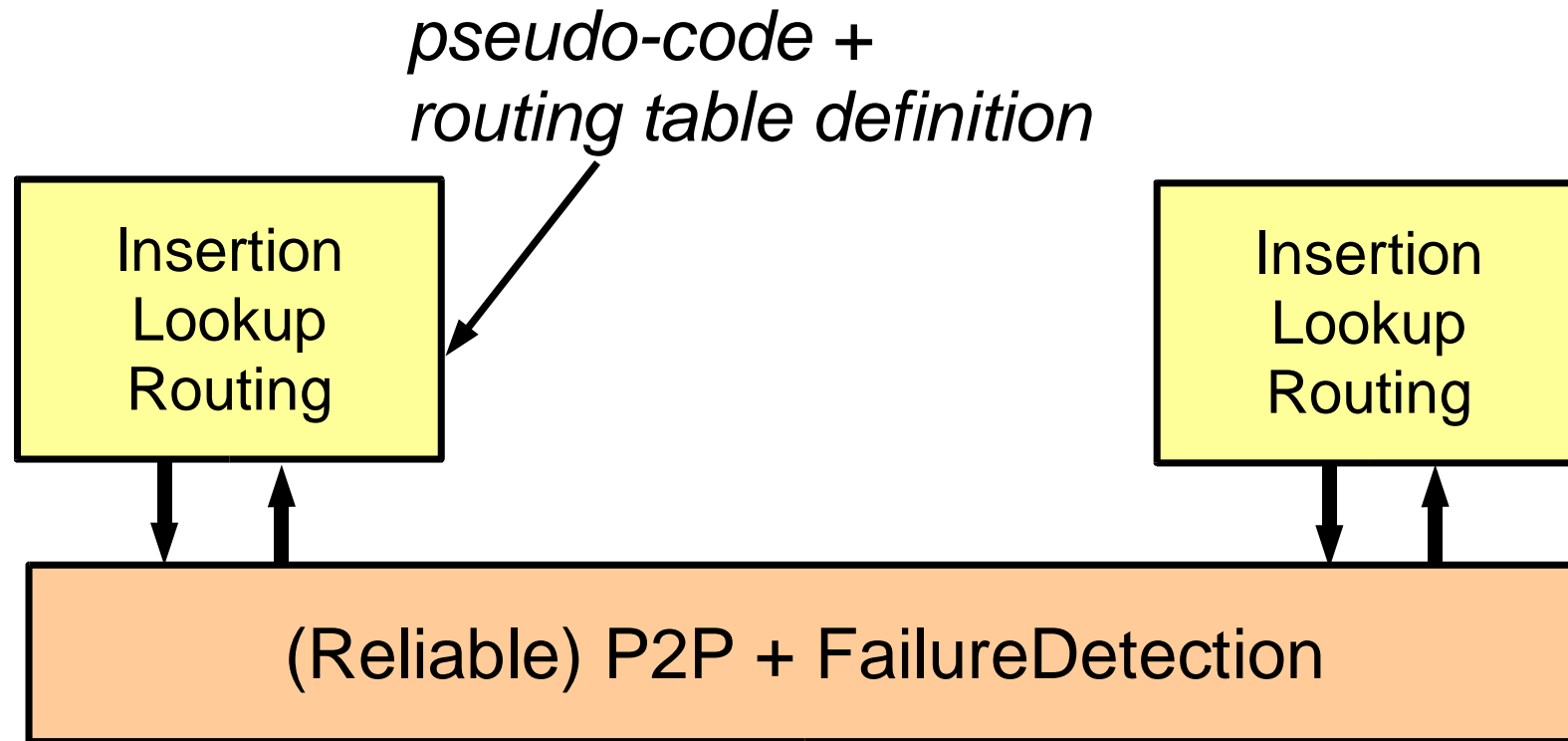
distributed group of processes provide a *hash table*
each process holds a *share* of the *data*



each process holds information about

- **this information is likely to be incorrect !**
- **consistency** (w.r.t. some logical id-space)

Design of DHT Algorithms

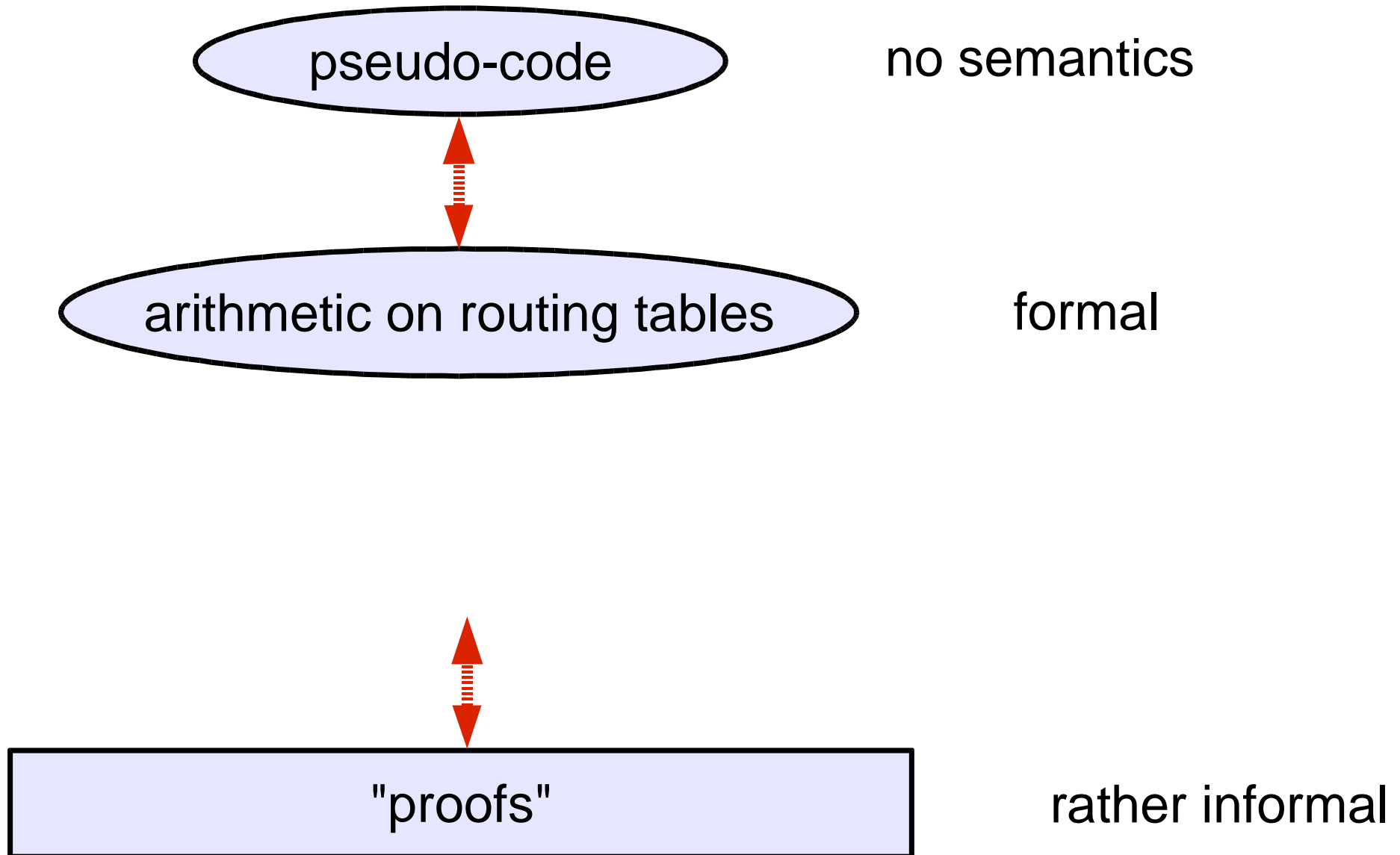


Consistency:

become probabilistic in case of joins/leaves !

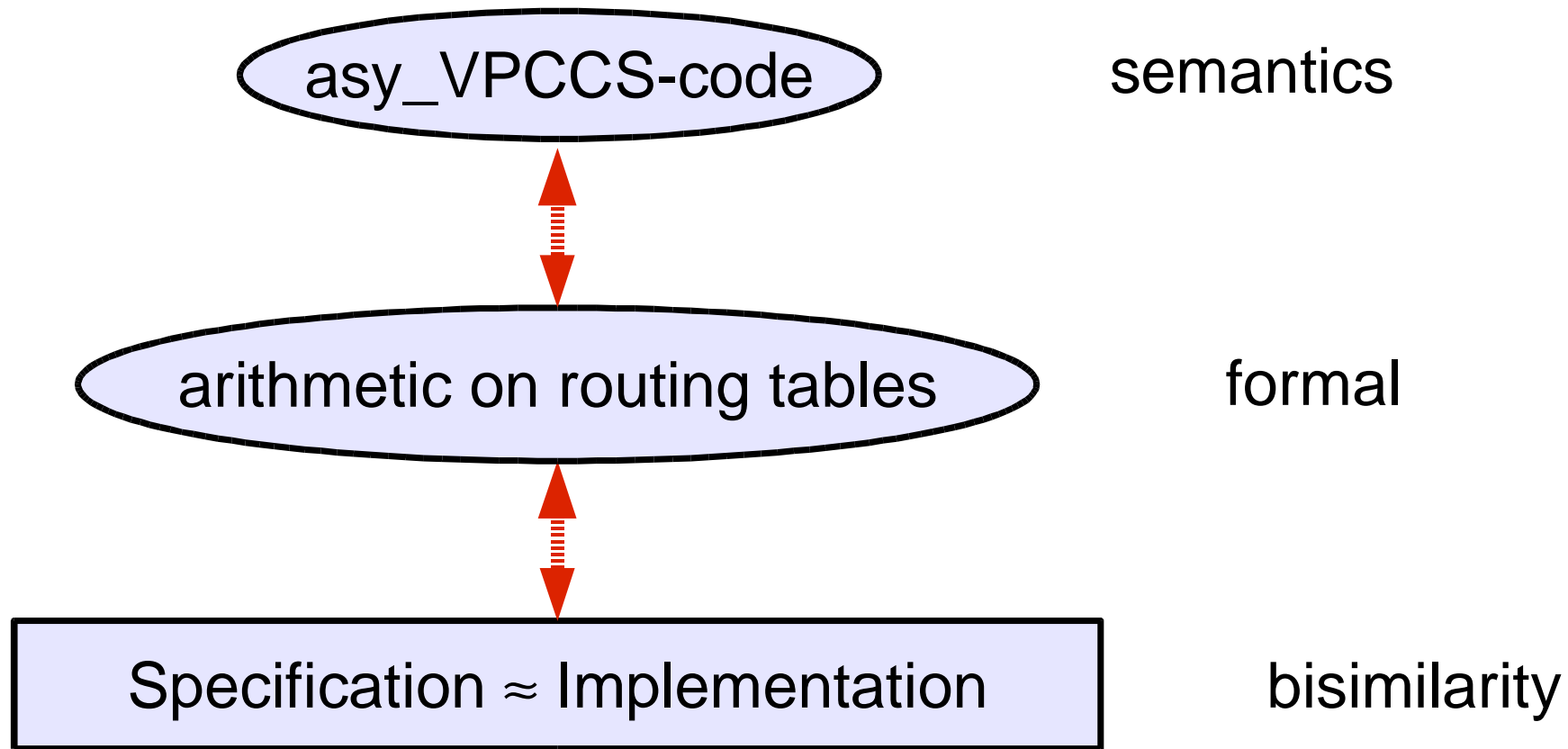
- self-stabilization of routing
- virtual consistency (related to database concepts)
- ...

Usually ...



This Year ... (Static Case)

no leaves, no joins, correct routing info everywhere



Soon? ... (Dynamic Case)

From Distributed to Global (I)

differences:

- *size matters ... for efficiency*
#nodes in the system: 10^2 --> 10^6
- *communication topology*
connectivity: full --> sparse
- *failure detection*
QoS: LAN --> WAN
- *efficiency measurement*
#messages: deterministic --> probabilistic
- *correctness formulations*
guranatees: deterministic --> probabilistic

From Distributed to Global (II)

common aspects:

- access to the **global state** is useful
... no: it is needed
- close **interaction with algorithm designers** is useful
... no: it is needed !
- techniques of **concurrency theory** do help
- non-trivial case studies take time ... :-)

Conclusions?

- interesting & challenging SGUC-problems
- identify important applications for case studies
- talk to and collaborate with "the other side"
- don't be dogmatic about the formalisms to use

Trust? Belief?

In the context of our case studies not (yet) an issue.

All processes run the same (set of) program(s) ...

... and they run human-free :-)