Computing & Computer Science: Foundation, Evolution & Perspectives

Digital Culture

Andrea Omicini
andrea.omicini@unibo.it

Master in Business Management / Bologna Business School

Academic Year 2017/2018

Next in Line...

Computing is not Programming

Computer Science is a Science

- *computer science* is not programming in the same way as mathematics is not just calculating or measuring
- understanding computer science by programming is more or less like understanding math by using abacus: it may work for some aspects, hardly for all

? so, what is computer science?

Computer Science vs. Computer Engineering

Computer *scientists* and *engineers*...

In theory

- *computer engineers* ground their technical knowledge upon some solid foundations
- conceptually provided by *computer scientists*

Actually, it is more complex than that.

- since the core of computer science is a *rapidly evolving body* that is actively built by computer engineering
- and, computer scientists and engineers are not so easy to distinguish one from another

Foundational Questions

What is

- science?
- engineering?
- computer science?
- computer engineering / software engineering?
- computation?
- a machine?
- a computer?
- a system?
- a computational system?

What is Science?

E.g., from the Science Council

http://sciencecouncil.org/about-us/our-definition-of-science/

> *Science is the pursuit and application of knowledge and understanding of the natural and social world following a systematic methodology based on evidence*

- well played, not really surprising
- in the overall, for all the basic questions we have answers that require some deep thinking, nevertheless they more or less belong to our background
- so, first of all, we have to find out *which* one is *the* basic question for us here

Phenomena vs. Noumena

Phenomenon

http://www.britannica.com/topic/phenomenon-philosophy

> *Phenomenon*, *in philosophy, any object, fact, or occurrence perceived or observed. In general, phenomena are the objects of the senses (e.g., sights and sounds) as contrasted with what is apprehended by the intellect. . . .*

Noumenon

http://www.britannica.com/topic/noumenon

> *Noumenon*, *plural Noumena, in the philosophy of Immanuel Kant, the thing-in-itself (das Ding an sich) as opposed to what Kant called the phenomenon—the thing as it appears to an observer. . . .*

What is Computer Science?

"Is there such a thing as computer science, and if there is, what is it?"

Wherever there are phenomena, there can be a science to describe and explain those phenomena. Thus, the simplest (and correct) answer to "What is botany?" is, "Botany is the study of plants." And zoology is the study of animals, astronomy the study of stars, and so on. Phenomena breed sciences.

There are computers. Ergo, computer science is the study of computers. The phenomena surrounding computers are varied, complex, rich. [Newell et al., 1967]

What is the Object of Study of Computer Science? I

What if *computer science is the study of computers*, yet...

The term "computer" is not well defined, and its meaning will change with new developments. *[Newell et al., 1967]*

A science may bear a shifting object of study

The phenomena of all sciences change over time; the process of understanding assures that this will be the case. Astronomy did not originally include the study of interstellar gases; physics did not include radioactivity; psychology did not include the study of animal behavior. Mathematics was once defined as the "science of quantity." *[Newell et al., 1967]*

What is the Object of Study of Computer Science? II

Whatever a computer *is*, what does a computer *do*?

A computer computes

What is the Object of Study of Computer Science? III

More generally...

- a computer is a *machine* that computes
- *computer systems*, or computational systems are systems made of computers
- computers and computational systems produce the *evidence*, the *facts*, the phenomena that are studied by computer science
- computation is then a core *object of study* of computer science
- the *essence* of *what computation is* represents then the noumenon at the core of computer science

What is the Object of Study of Computer Science? IV

Definitions of *computation* and *computer science* go hand in hand

Over time, the definition of computer science has been a moving target. These stages reflect increasingly sophisticated understandings of computation. *[Denning, 2010]*

What is the Object of Study of Computer Science? V

"The history of computer science reveals an interesting progression of *definitions* for computer science" [Denning, 2008]

- study of automatic computing (1940s)
- study of information processing (1950s)
- study of phenomena surrounding *computers* (1960s)
- study of *what* can be automated (1970s)
- study of computation (1980s)
- study of information processes, both *natural* and *artificial* (2000s)

Next in Line...

What is Computation? I

Question "What is Computation?" considered as harmful [Freeman, 2011]

It is important to have a common understanding of fundamentals in order to make progress in any field, but a rigid "standard" that is adopted too early is almost always an impediment to progress. Just think of where we would be today if computer science had remained merely a branch of mathematics or engineering or experiment-based science.

... said that, *"What is computation?"* is the basic question

- around which all Computer Science revolves
- where any well-founded study of nowadays computational systems should be grounded on

What is Computation? II

Computation is *symbol manipulation*

A computation is a sequence of simple, well-defined steps that lead to the solution of a problem. The problem itself must be defined exactly and unambiguously, and each step in the computation that solves the problem must be described in very specific terms. [Conery, 2010a]

. . . a problem, and its solution, must be encoded in the form of symbols; a step is a symbol manipulation that transforms one set of symbols into a new set of symbols [Conery, 2010b]

What is Computation? III

Computation is *process*

> *. . . the essence of computation can be found in any form of process* *[Frailey, 2010]*

→ so, computation is a *process*, and every process is also a computation [Frailey, 2010]

What is Computation? IV

Process

`http://www.oxforddictionaries.com/definition/english/process`

process,

1. *A series of actions or steps taken in order to achieve a particular end. . .*
    1. *A natural series of changes. . .*
    2. *A systematic series of mechanized or chemical operations that are performed in order to produce something. . .*
    3. *An instance of a program being executed in a multitasking operating system, typically running in an environment that protects it from other processes. . .*

What is Computation? V

When defining computation... [Denning, 2011]

- computational model matters
- many important computations are natural
- many important computations are non-terminating
- many important computations are continuous
- computational thinking can be defined

Computational Model

Computing machines

- Turing's computing machine [Turing, 1937]
- von Neumann's computing machine [Burks et al., 1982]

? they are *artificial*, do they work when we include *natural* computations?

? they are *discrete*, do they work when we include *continuous* computations?

? they represent one *single* computing device, do they work when we deal with *computational systems*?

Computational Model

Computing machines

- Turing's computing machine [Turing, 1937]
- von Neumann's computer machine [Burks et al., 1982]

? they are *artificial*, do they work when we include *natural* computations?

? they are *discrete*, do they work when we include *continuous* computations?

? they represent one *single* computing device, do they work when we deal with *computational systems*?

What is a Machine? I

Machine

http://www.britannica.com/technology/machine

Machine, device, having a unique purpose, that augments or replaces human or animal effort for the accomplishment of physical tasks. . . . The operation of a machine may involve the transformation of chemical, thermal, electrical, or nuclear energy into mechanical energy, or vice versa, or its function may simply be to modify and transmit forces and motions. All machines have an input, an output, and a transforming or modifying and transmitting device. . . .

What is a Machine? II

Input, output & state of a machine

- input is what affects a machine from the outside
- output is how a machine affects the outside
- state at time t is whatever is necessary to understand the evolution of a machine after t given some input—or, more generally, given the *context* where the machine operates

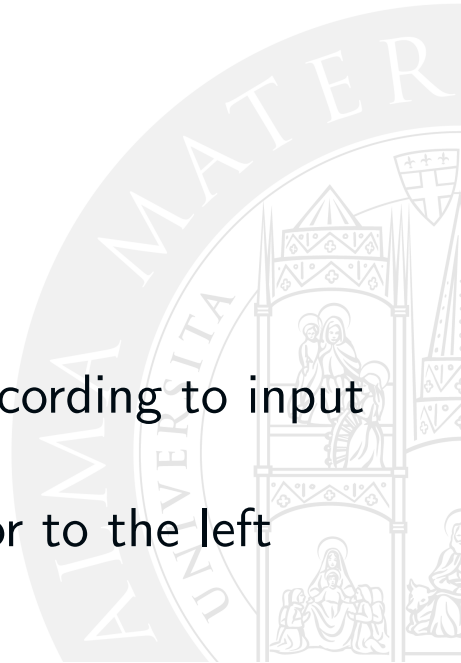
What is a Computing Machine? *(to be completed)*

A computing machine is a different sort of machine...

- whose task is *symbolic* / abstract / cognitive instead of *physical*
- whose input and output are basically *information*—in some form
- whose *context* is...?

Turing Machine I

- a Turing machine is an *abstract machine*
  - it is a *theoretical* machine since it cannot be built
- notions like *computability* and *computational complexity* can be defined based on it
- a simple Turing machine consists of
  - an infinite *tape* (to the right, if starting for left)
  - made of *cells*, or sections on the tape
  - each cell holding either a *symbol* or blank
  - symbols are the machine *input* and *output*
  - the machine has a *state*
  - the *head* read a symbol from the tape, and behave according to input and state
  - by writing a symbol, then moving either to the right or to the left



Turing Machine II

Church-Turing thesis

- a function can be computed using an *algorithm* if and only if is computable by a Turing machine
- where an algorithm is basically is a (non-ambiguous) specification (in some language) of the solution for a class of problems
- it is a *thesis* because it is not really *proven*

Computability

- the notion of computability defined by Church-Turing thesis is so effective that nowadays *computability* and *computability by Turing machine* are basically synonym
- there are *computable* and *non-computable* functions—e.g., Busy Beaver function, or, the *halting problem*

Turing Machine III

Expressiveness

- different models of computation can be adopted, then different abstract machines
- and different *programming languages* over them—e,g, Java over the Java VM
- expressiveness of (computational) programming languages is measured in terms of Turing equivalence

Turing Machine IV

Decidability

- a Turing machine cannot always respond yes or no when asked about the termination of an *arbitrary computer program*
- the so-called *halting problem* is not decidable
- and so are a class of (non-decidable) problems that cannot be decided by (Turing-equivalent) computing

Turing Machine V

Complexity

- computational models can be used to *classify problems* according to the inherent *difficulty*
- using computational models – such as the Turing machine – to measure the *time* and *space* required to solve the problems themselves
- complexity classes of problems give us the basic notion of the resources required to solve a problem computationally

Von Neumann Machine I

Basic computer architecture

- an *architecture* for a digital computer, made of
  - a (central) *processing unit* (CPU), including ALU (*arithmetic logic unit*) and *registers*
  - a *control unit*, including *instruction register* and *program counter*
  - a *memory* to store both *data* and *instructions*
  - *external* mass *storage*
  - *input* / *output*
  - a connecting *bus*

Von Neumann Machine II

Basic working cycle

- *fetch*, *decode*, *execute* cycle
  1. fetch instruction from memory
  2. fetch data required by the instruction from memory
  3. execute the instruction / process the data
  4. store results in memory
  5. jump to step 1

Von Neumann Machine III

Sequential computation & control flow

- current computers can still be interpreted by exploiting some of the early ideas from Von Neumann architecture
- some basic notions are easy to identify there

  sequential computation as the process of executing one instruction at a time, one after another

  control flow as the sequence of points in the program identified by the program counter

What is a Computing Machine? *(now complete)*

A computing machine is a different sort of machine...

- whose task is *symbolic* / abstract / cognitive instead of *physical*
- whose input and output are basically *information*—in some form
- whose *context* is...?

Abstracting away from (computing) machinery

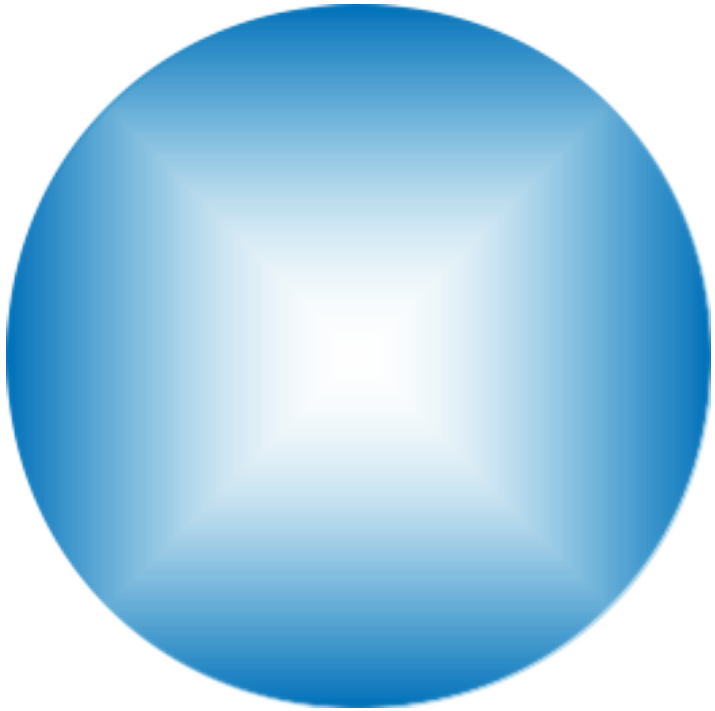
- if we choose not to stick with one specific computational model, it might be appropriate to abstract away from the *machinery*
- by focussing instead on the *computational process*

Computational Process I

Assumptions

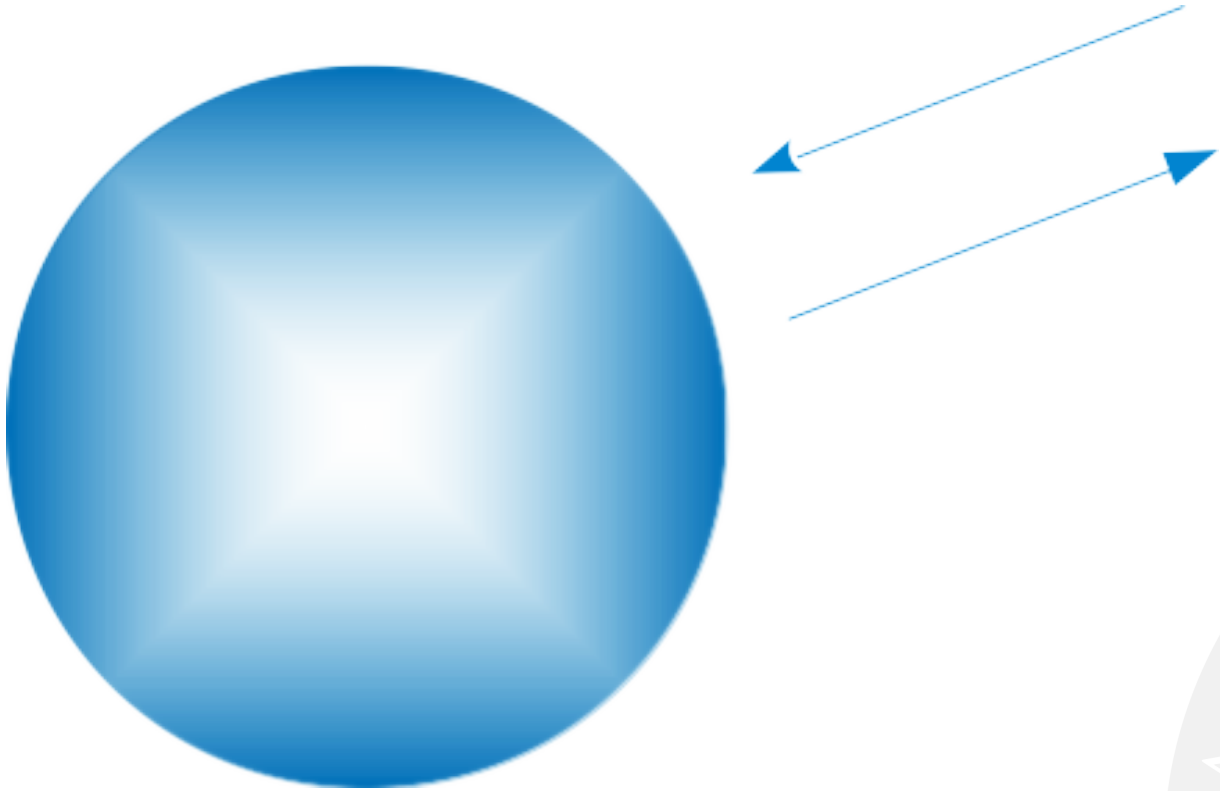
- the *elementary* computational process is sequential
- since it represent the *phenomenal* expression of the dynamics of a computing machine, it has both
  - *input / output*
  - *context*
- as a result, in the following a *computing machine* is the place where a *computational process* occurs

Computational Process II



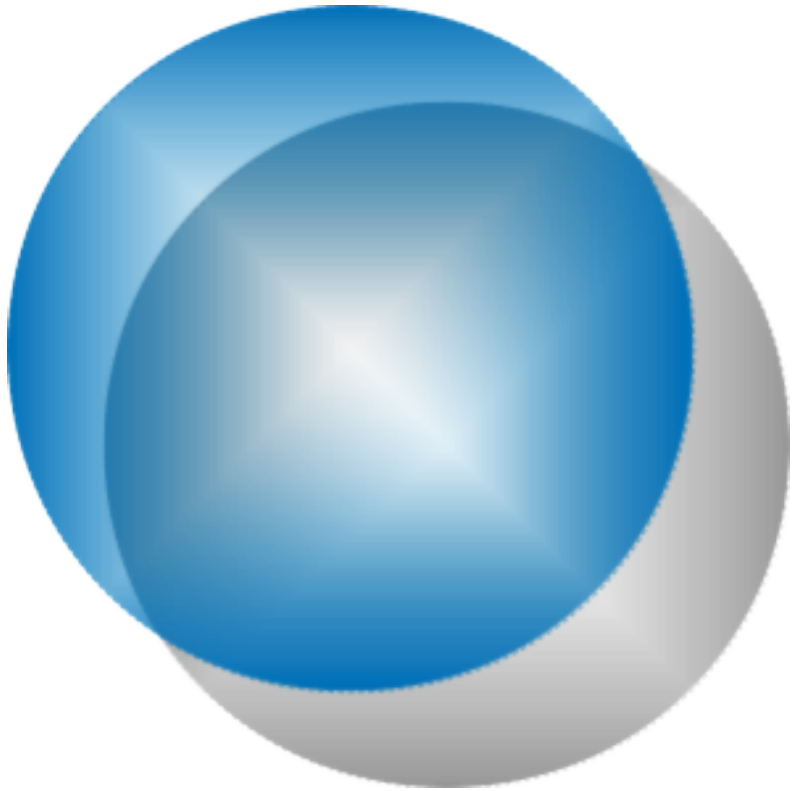
Our representation for a computational process:
sequential computation occur inside the blue circle

Computational Process III



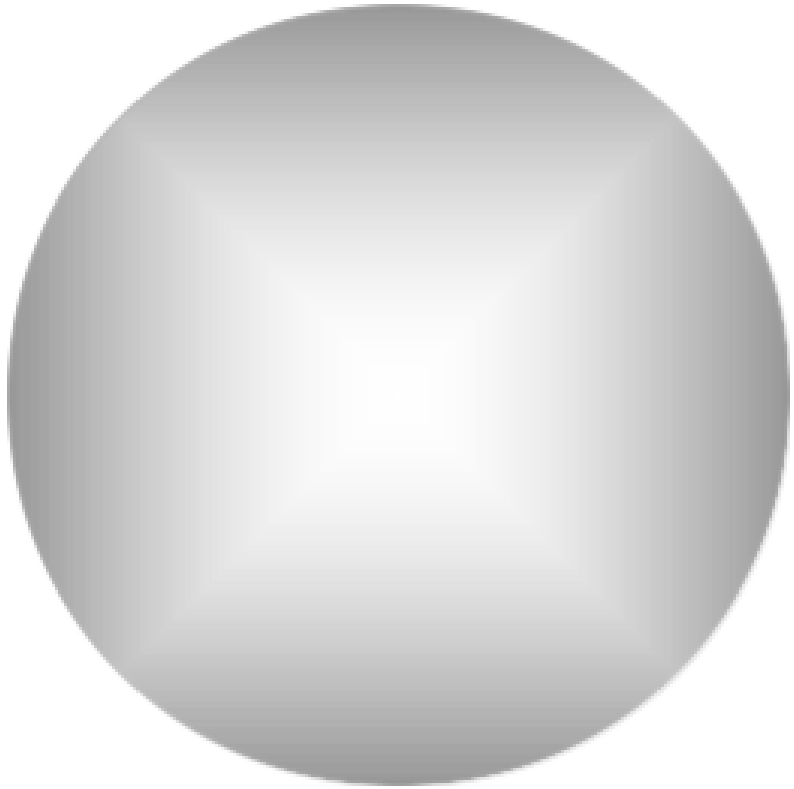
Computational process with *input* and *output*

Computational Process IV



Computational process with *context*

Context for Computation I



Computational *context*

Context for Computation II

What is context when computation is concerned?

- *computing machine*
- resources
- time
- space

Context for Computation III

When do we need to *represent* context for computation?

Whenever either

- computing machine
- resources
- time
- space

or, all of them, are *relevant* to model / represent / understand computation—that is,

- *understanding the dynamics* of the computational process

Context for Computation IV

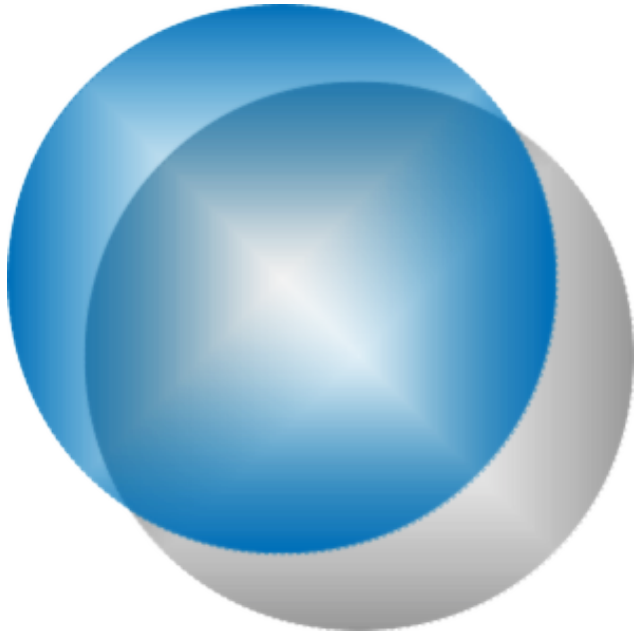
Sorts of computations

- timed computation, whenever the *time* of the computational machine is relevant / essential for the computing process
- spatial computation, whenever the *spatial features* of the computational machine are relevant / essential for the computing process
- more generally, situated computation [Suchman, 1987], whenever the *environment* of the computational machine is relevant / essential for the computing process
  - where the environment is any meaningful combination of temporal and spatial features with the *resources* required by the computation

Context for Computation V

Representing context for computation

- so, understanding a computational process requires the precise definition of its computational context
- graphically, a computational process (*blue area*) depends on how we define the features of the context (*grey area*)



Next in Line...

What is a System?

System

http://www.oxfordlearnersdictionaries.com/definition/english/system

> *. . . a group of things, pieces of equipment, etc. that are connected or work together . . .*

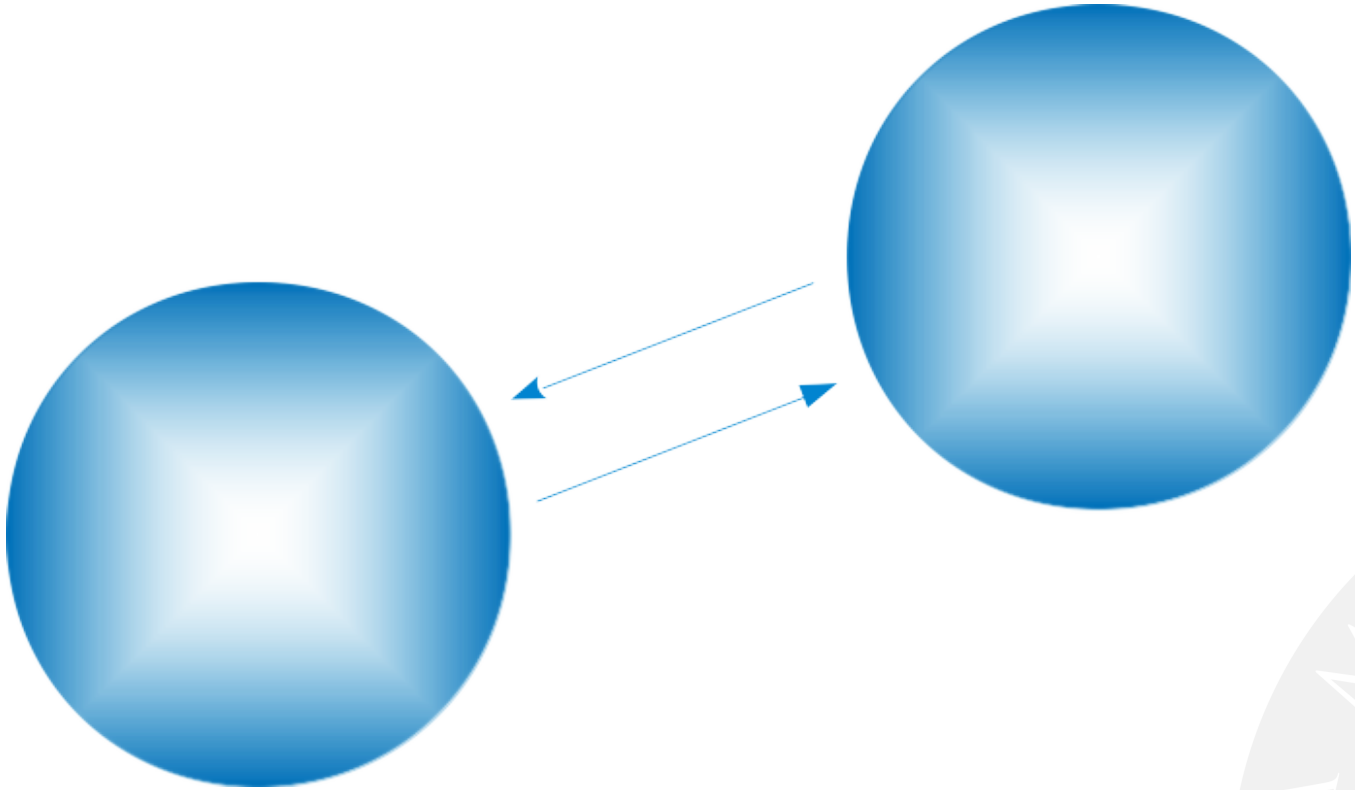
(Interacting) Computational System [Goldin et al., 2006] I

Computational system

In a computational system, two or more computational processes

- *behave* (by computing), and
- *work together* (by interacting)

(Interacting) Computational System [Goldin et al., 2006] II



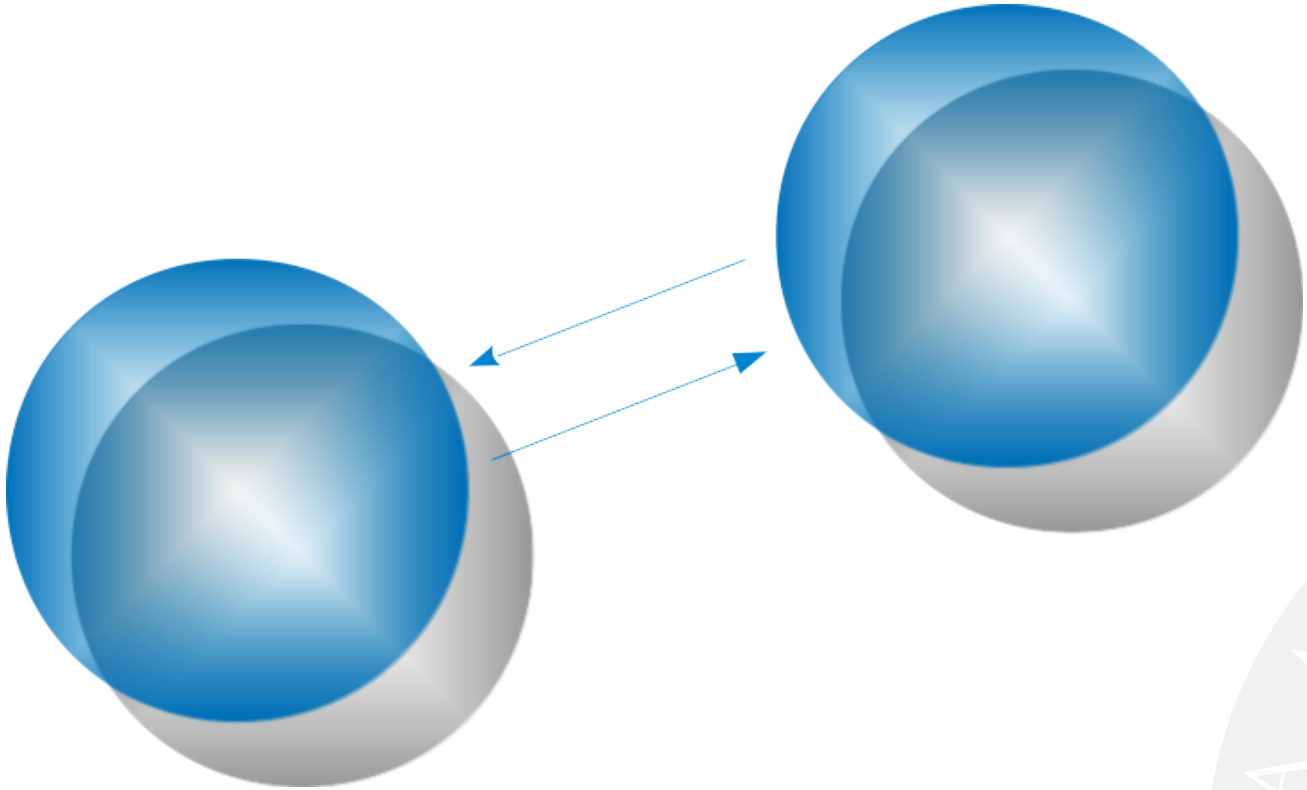
Computational *system*

What is Context for a Computational System? I

How should we represent the context for a computational system?

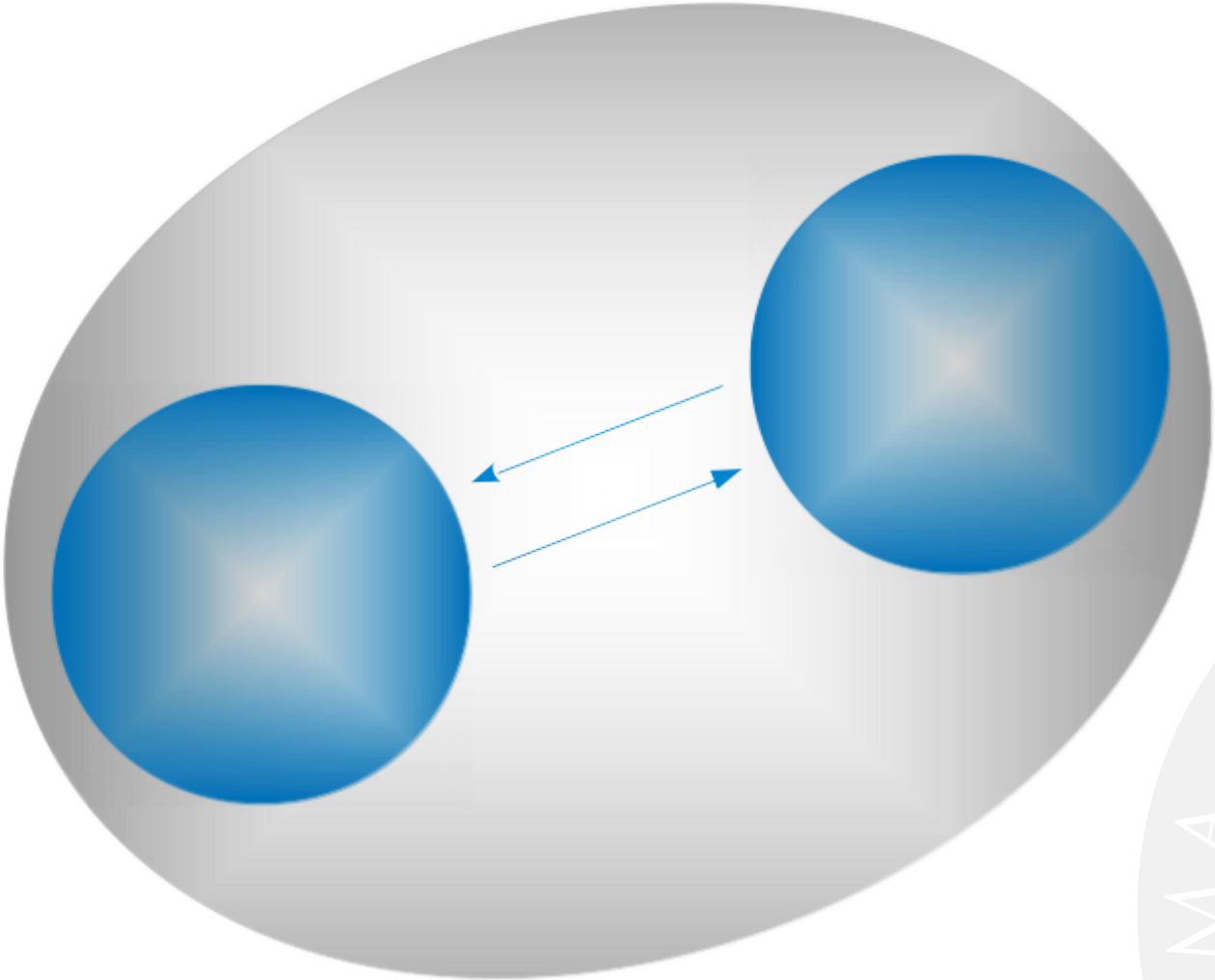
- as one separate, different context for each computational process?
- as a single context for the overall computational system?
- as a combination of the above choices?

What is Context for a Computational System? II



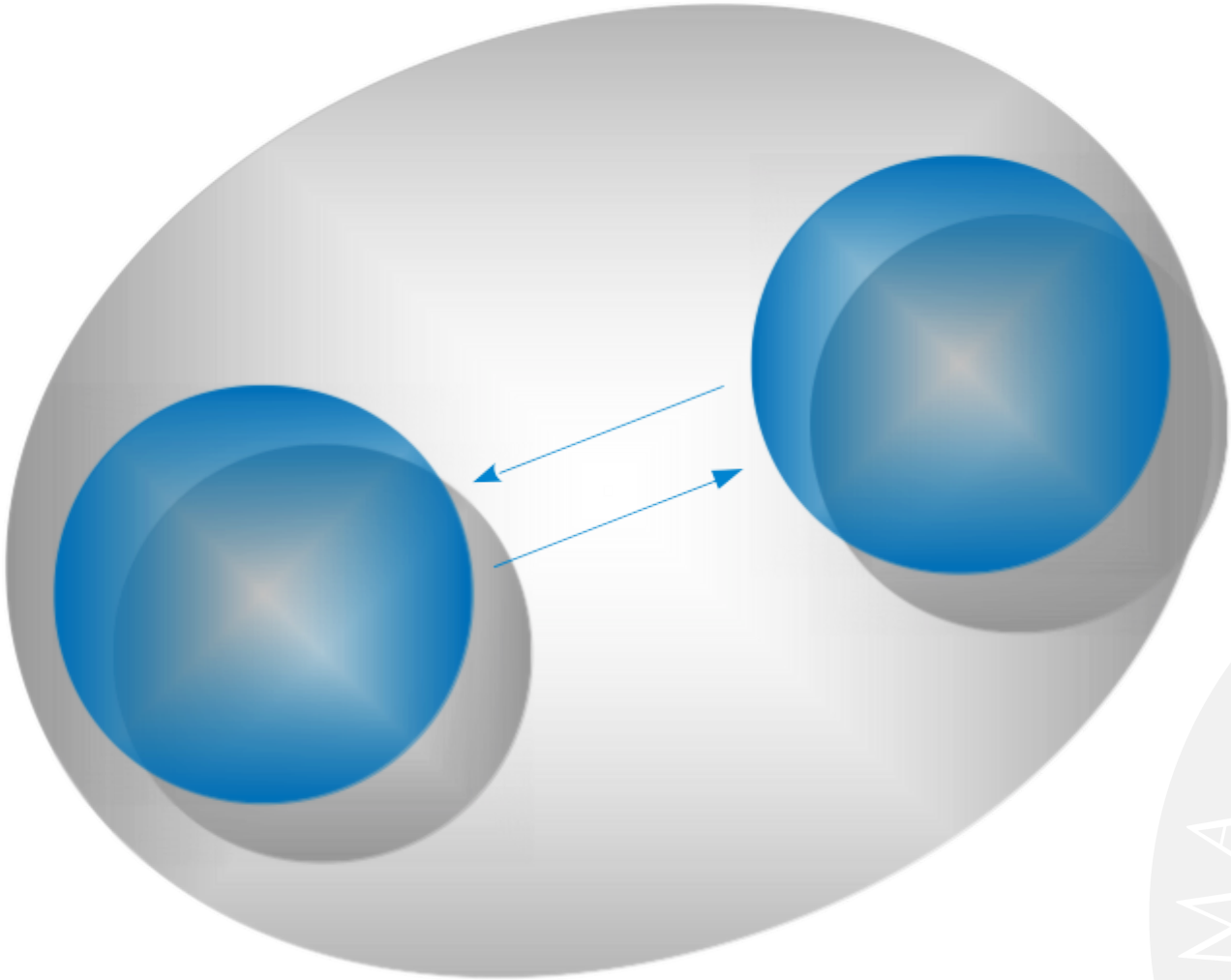
A different context for each process

What is Context for a Computational System? III



One context for all processes

What is Context for a Computational System? IV



A different context for each process plus one context for all processes

What is Context for a Computational System? V

Different contexts, different sorts of systems

The choice of the sort of the context defines the sort of the computational system, such as

- parallel systems
- concurrent systems
- distributed systems

Which Abstract Machines for (Interacting) Computational Systems?

How should we model computations in a computational system?

- just composing Turing / Von Neumann machines?
- what about the *interaction*?
- how to we model interaction through the *network*?
- what about expressiveness, and computational complexity, and so on?

Complexity & Interaction I

An essential source of complexity for computational systems is

interaction

[Goldin et al., 2006]

The power of interaction [Wegner, 1997]

Interaction is a more powerful paradigm than rule-based algorithms for computer-based solving, overtiring the prevailing view that all computing is expressible as algorithms.

Complexity & Interaction II

Intelligence & interaction [Brooks, 1991]

Real computational systems are not rational agents that take inputs, compute logically, and produce outputs. . . It is hard to draw the line at what is intelligence and what is environmental interaction. In a sense, it does not really matter which is which, as all intelligent systems must be situated in some world or other if they are to be useful entities.

A conceptual framework for interaction [Milner, 1993]

. . . a theory of concurrency and interaction requires a new conceptual framework, not just a refinement of what we find natural for sequential [algorithmic] computing.

Complexity & Interaction III

Interactive computing [Wegner and Goldin, 1999]

- *finite computing agents* that interact with an environment are shown to be more expressive than Turing machines according to a notion of expressiveness that measures problem-solving ability and is specified by observation equivalence
- *sequential interactive models* of objects, agents, and embedded systems are shown to be more expressive than algorithms
- *multi-agent* (distributed) *models of coordination*, collaboration, and true concurrency are shown to be more expressive than sequential models

Complexity & Interaction IV

Basically, where does complexity come from?

- events in a sequential component are *totally ordered*
- as soon as we combine components in a concurrent system (*distribution in time*), they are *no* longer totally *ordered*
- as soon as we combine components in a distributed system (*distribution in space*), interaction occurs in different contexts
- interaction make the overall system essentially unpredictable

? why is this *not a problem*?

! the range of behaviours that an interactive system can exhibit is typically larger than non-interactive systems

→ more behaviours means more expressiveness

Parallel vs. Concurrent Systems I

Parallel computing in the literature

- the term is typically used for non-sequential computing processes, where more than one computation can be performed at the same time

  [Shonkwiler and Lefton, 2006]
- typically requires *multi-core* architectures
- usually exploited to solve *computationally-intensive* scientific / mathematical problems

Parallel vs. Concurrent Systems II

Concurrency in the literature

- the term is typically used with a twofold acceptation
  [Degano and Montanari, 1987]
  - *interleaving*, where events occurring in separate concurrent processes could occur in *any* relative *order*—temporal / causal relations between events are not relevant
  - *true concurrency*, where *partial orderings* are used to explicitly capture temporal / causal relations between events

Parallel vs. Concurrent Systems III

Concurrency vs. parallelism

- relative ordering of events is the main point here
  - in parallel systems, events are *totally* ordered
    - in the same way as in sequential processes
  - in concurrent systems, events are at most *partially* ordered
- *temporal* relation

→ *temporal* context sets the difference

Distributed Computing & Systems I

Distributed computing in the literature

- the term typically refers to a number of asynchronous computational processes *located* on *different* devices and communicating via message passing (no shared memory) [Kshemkalyani and Singhal, 2011]

  Distributed computing is an activity that is performed on a spatially distributed system *[Lamport and Lynch, 1990]*

Distributed Computing & Systems II

Distributed systems in the literature

- the term typically refer to a collection of devices working together through a network connection

  A distributed system is a collection of independent computers that appears to its users as a single coherent system

  [Tanenbaum and van Steen, 2007]

Distributed Computing & Systems III

Distribution

- *physical distribution* of computational processes and computing devices is the main point here
  - in distributed computing, the focus is on the spatial distribution of processes
  - in distributed systems, the focus is on the spatial distribution of devices
- *spatial* relation

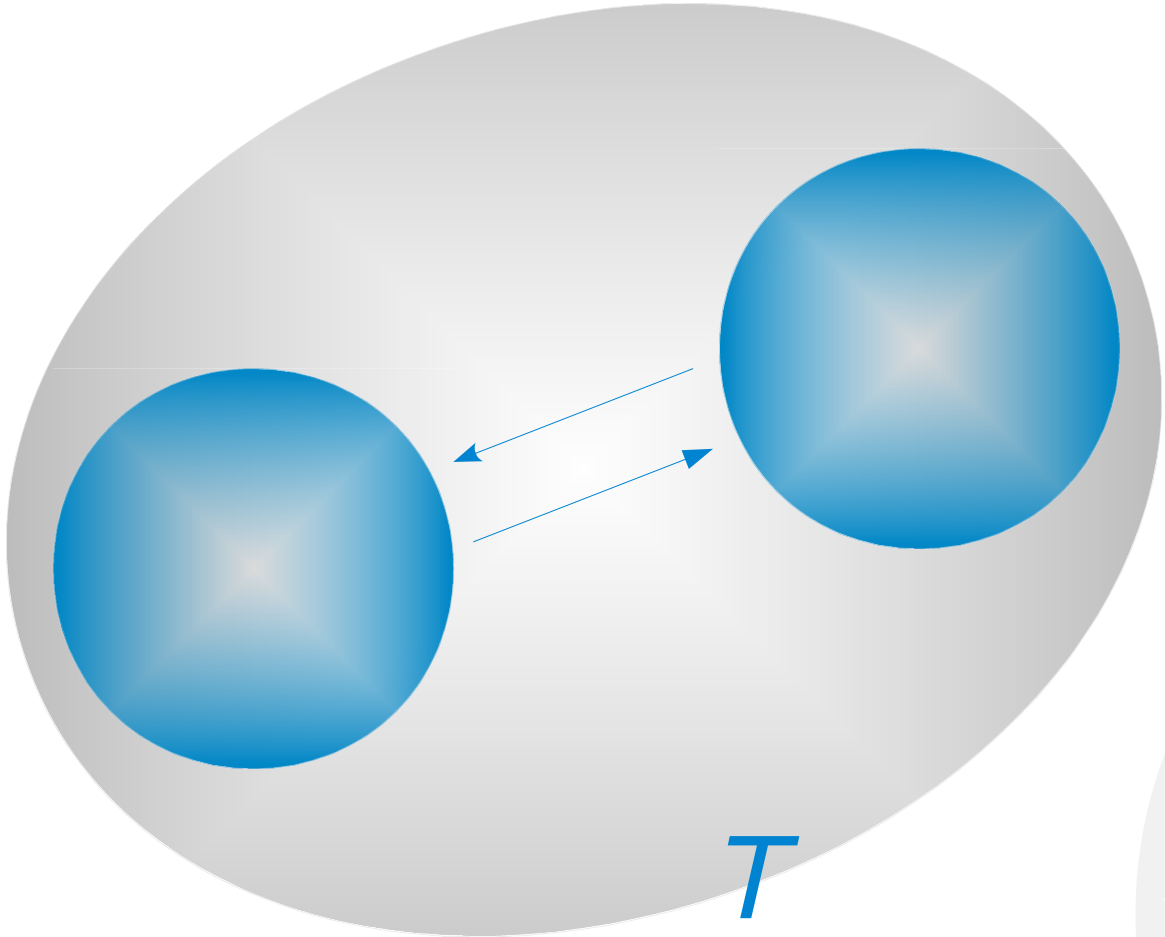
$\rightarrow$ *spatial* context defines both

Definitions I

Parallel computing & systems

- given a computational system, we talk of parallel computation whenever the *temporal* context is the same for all computational process
- a parallel system is a computational system performing parallel computations

Definitions II



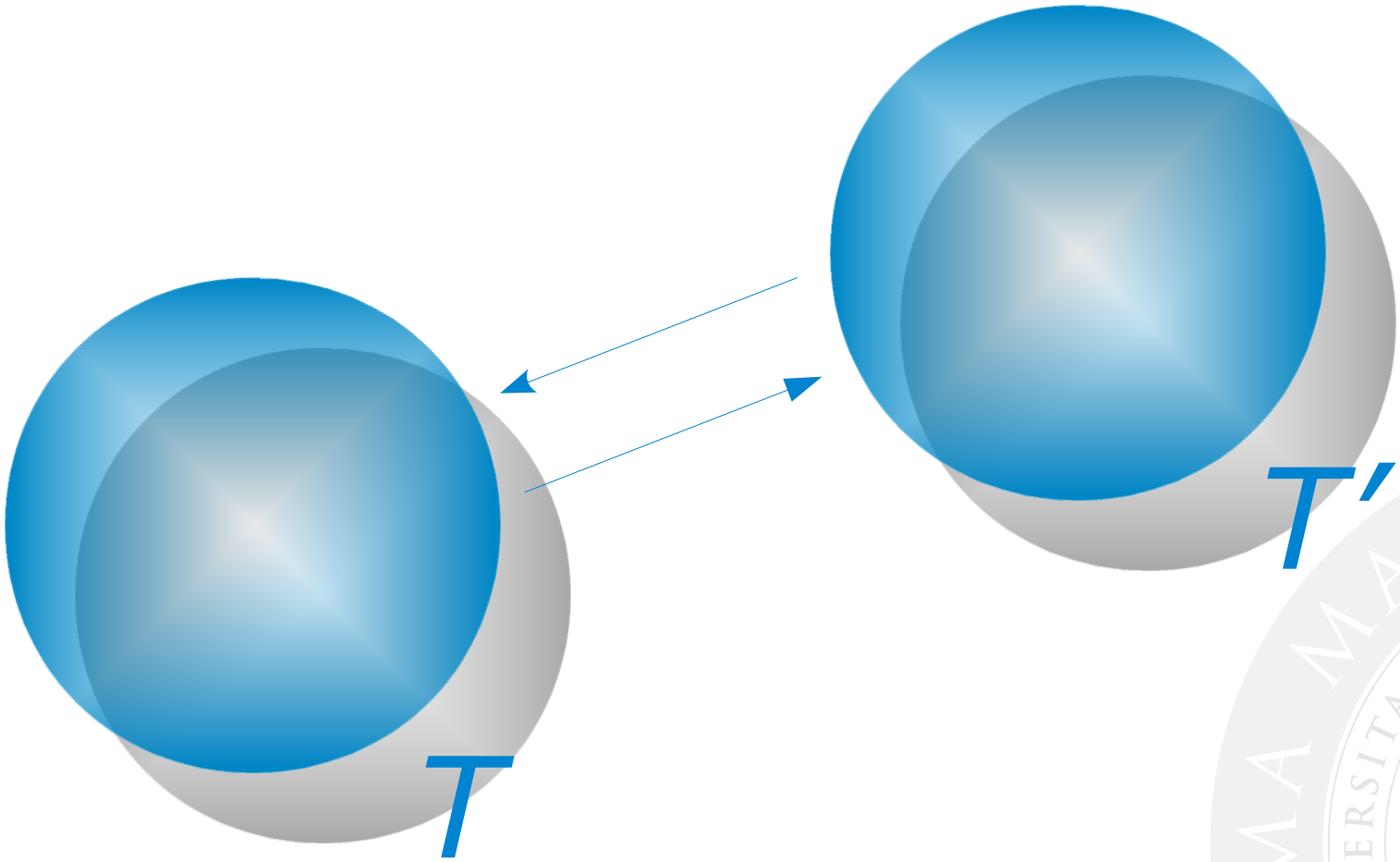
Parallel computing: the same temporal context T for all processes

Definitions III

Concurrent computing & systems

- given a computational system, we talk of concurrent computation whenever at least two computational processes have a different *temporal* context
- a concurrent system is a computational system performing concurrent computations

Definitions IV



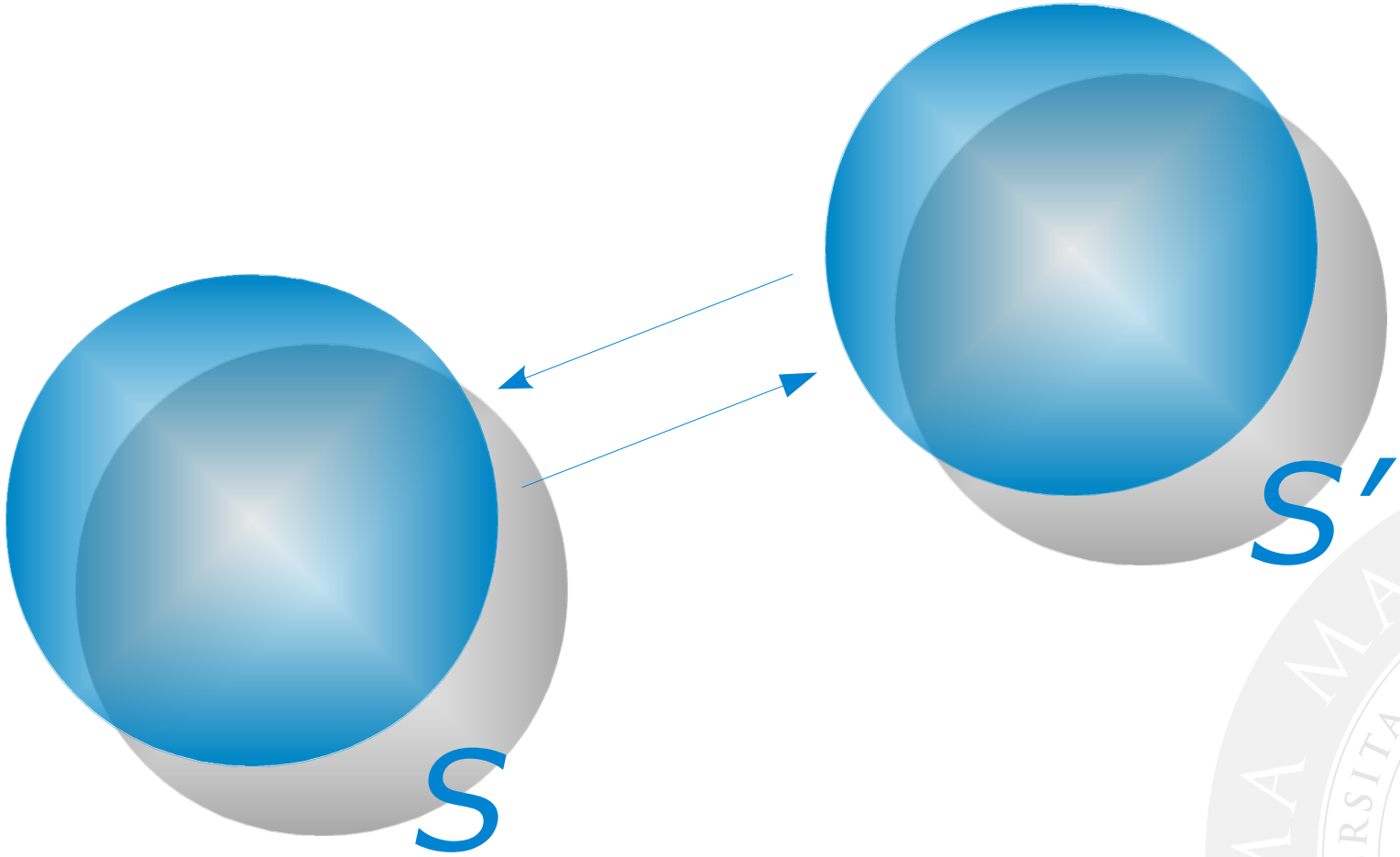
Concurrent computing: different temporal contexts $T \neq T'$ for different processes

Definitions V

Distributed computing & systems

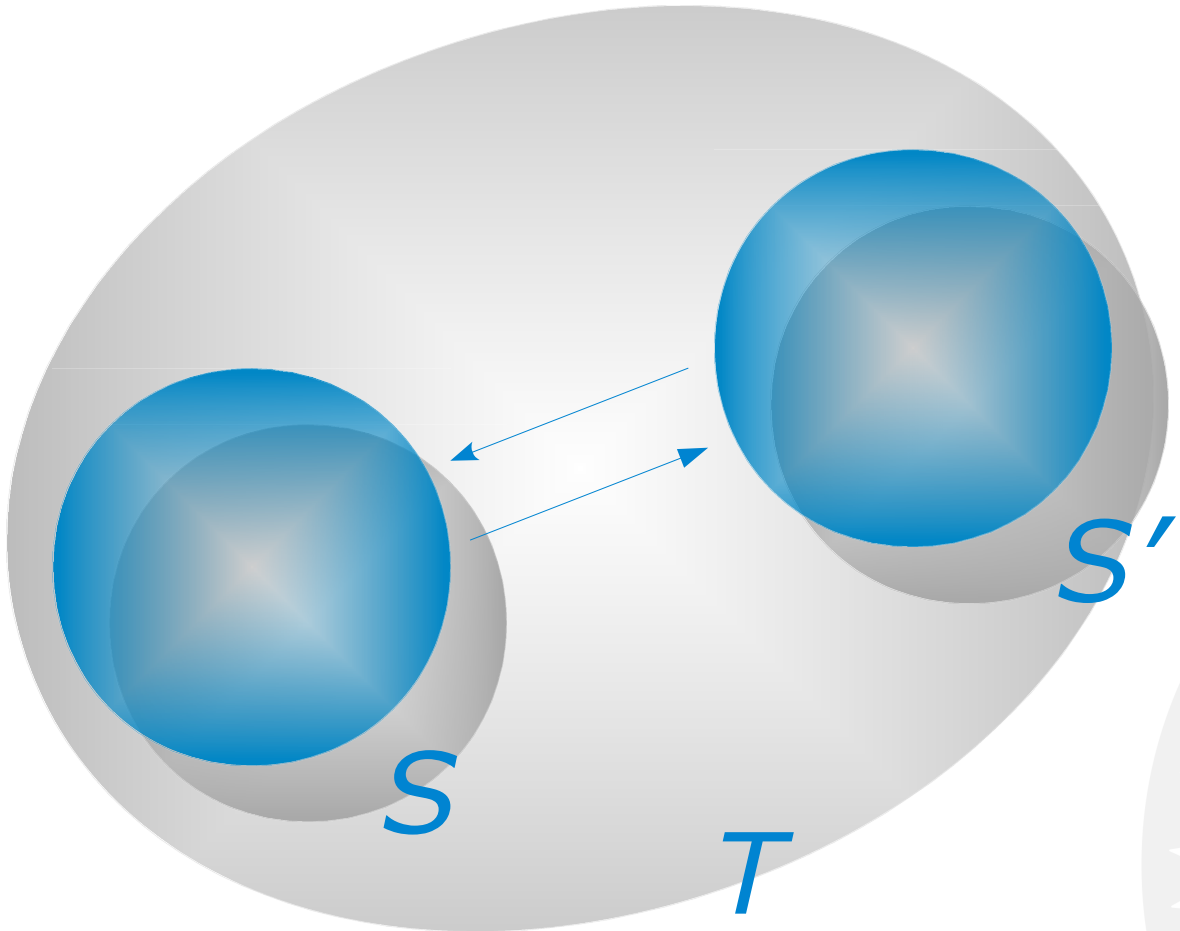
- given a computational system, we talk of distributed computation whenever at least two computational processes have a different *spatial* context
- a distributed system is a computational system performing distributed computations

Definitions VI



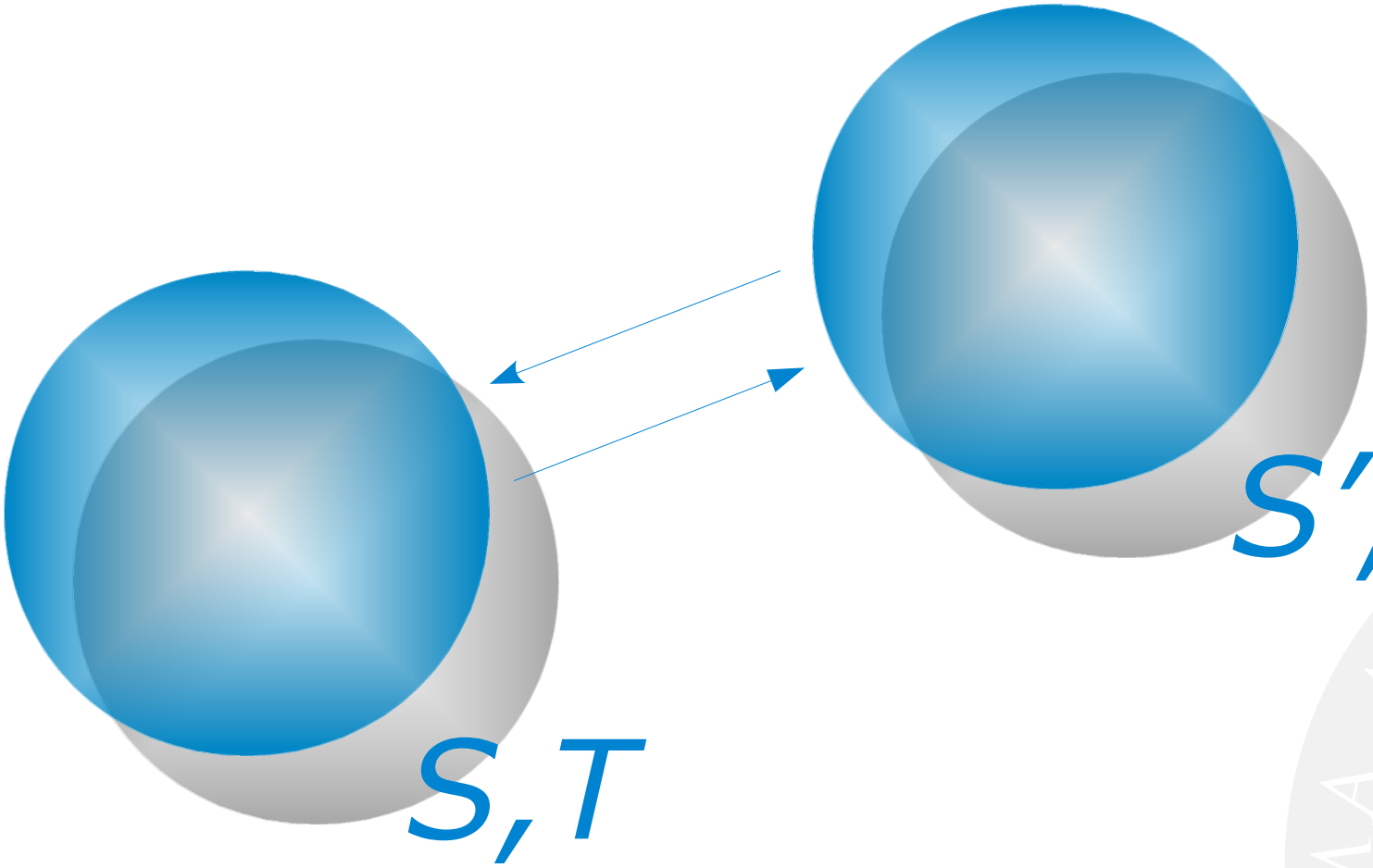
Distributed computing: different spatial contexts $S \neq S'$ for different processes

Spatial vs. Temporal Contexts I



Distributed parallel computing: $S \neq S'$, same T

Spatial vs. Temporal Contexts II



Distributed concurrent computing: $S \neq S'$, $T \neq T'$

Next in Line...

Evolution of Distributed Systems I

Evolving technological and application scenarios...

- ... forced new requirements, structure, and goals for computational systems
- making distributed systems the default for nowadays computational systems
- resulting in the emergence of diverse *classes of distributed systems* over time

Evolution of Distributed Systems II

Diverse sorts of distributed systems

- roughly telling the story of the *evolution* of distributed systems over the years
- depending on both the *environment* where they are developed, the *goals* they have to achieve, and the level of the available *technologies*
- different *models*, *methodologies*, and *technologies* are to be used to design and develop different sorts of distributed systems

Sorts of Distributed Systems

Three main classes of distributed systems

- *distributed computing systems*
- *distributed information systems*
- distributed *pervasive systems*

Distributed Computing Systems

The main characteristic

- using a *multiplicity of distributed computers* to perform *high-performance* tasks

Two classes

- cluster computing systems
- grid computing systems

Cluster Computing Systems I

The basic idea

- a collection of similar workstations / PCs
- running the same OS
- located in the same area
- interconnected through a high-speed LAN

Motivation

- the ever increasing price / performance ration of computers makes it cheaper to build a supercomputer by putting together many simple computers, rather than buying a high-performance one
- also, robustness is higher, maintenance and incremental addition of computing power is easier

Cluster Computing Systems II

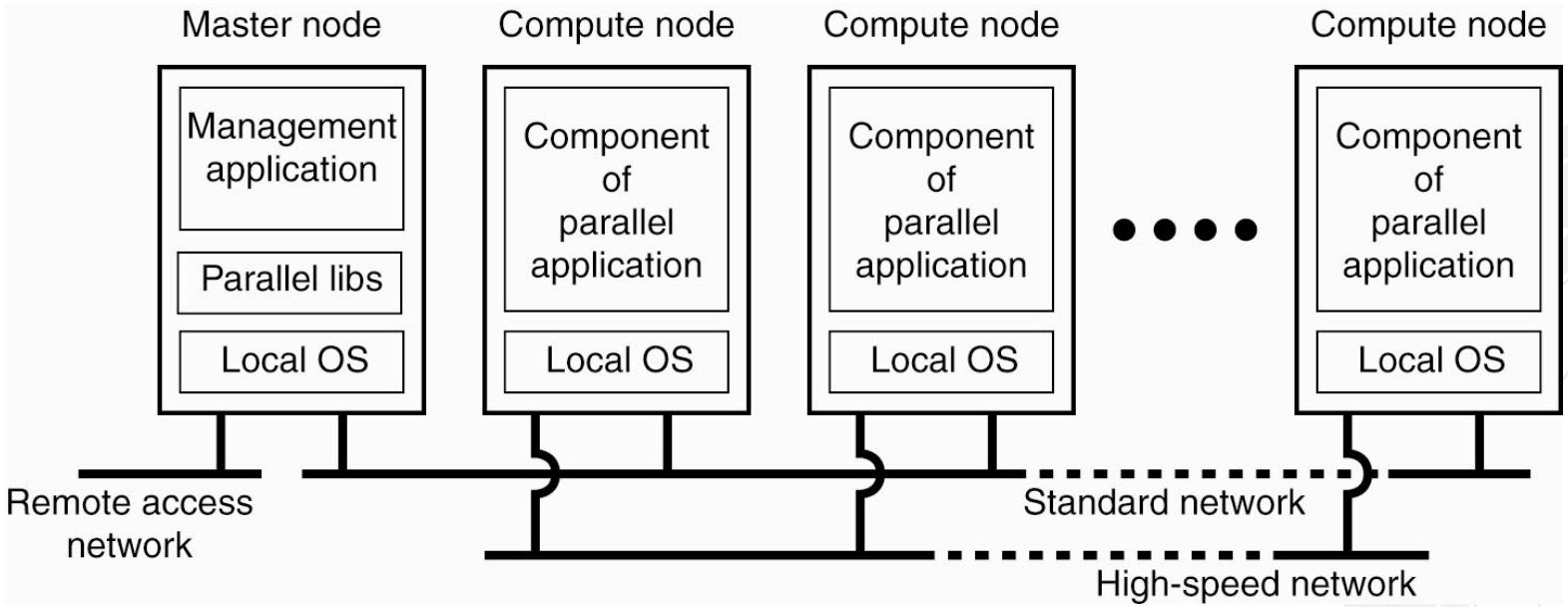
Usage

- parallel computing
- typically, a single computationally-intensive program is run in parallel on multiple machines

Cluster Computing Systems: Example

Beowulf clusters

- Linux-based
- each cluster is a collection of computing nodes controlled and accessed by a *single* master *node*



[Tanenbaum and van Steen, 2007]

Cluster vs. Grid Computing Systems

Homogeneity vs. heterogeneity

- homogeneity
  - computers in a cluster are typically similar
  - computers in a cluster have the same OS
  - computers in a cluster are connected to the same (local) network
- in essence, cluster computer systems are homogeneous
- grid computer systems instead are typically heterogeneous

Grid Computing Systems

The main idea

- resources from different organisations are brought together to promote *collaboration* between individuals, groups, or institutions, by passing organisation boundaries
- collaboration is built in the form of a *virtual organisation*
  - essentially, a new virtual organisational entity including people from existing organisations
  - accessing resources made available by participating organisations
  - including servers, databases, hard disks, ...
- by their very nature, grid computer systems deal with different administrative domains

Architecture of a Grid Computing System I

A layered architecture for a grid computing system [Foster et al., 2001]

fabric layer — interface to local resources at a specific site

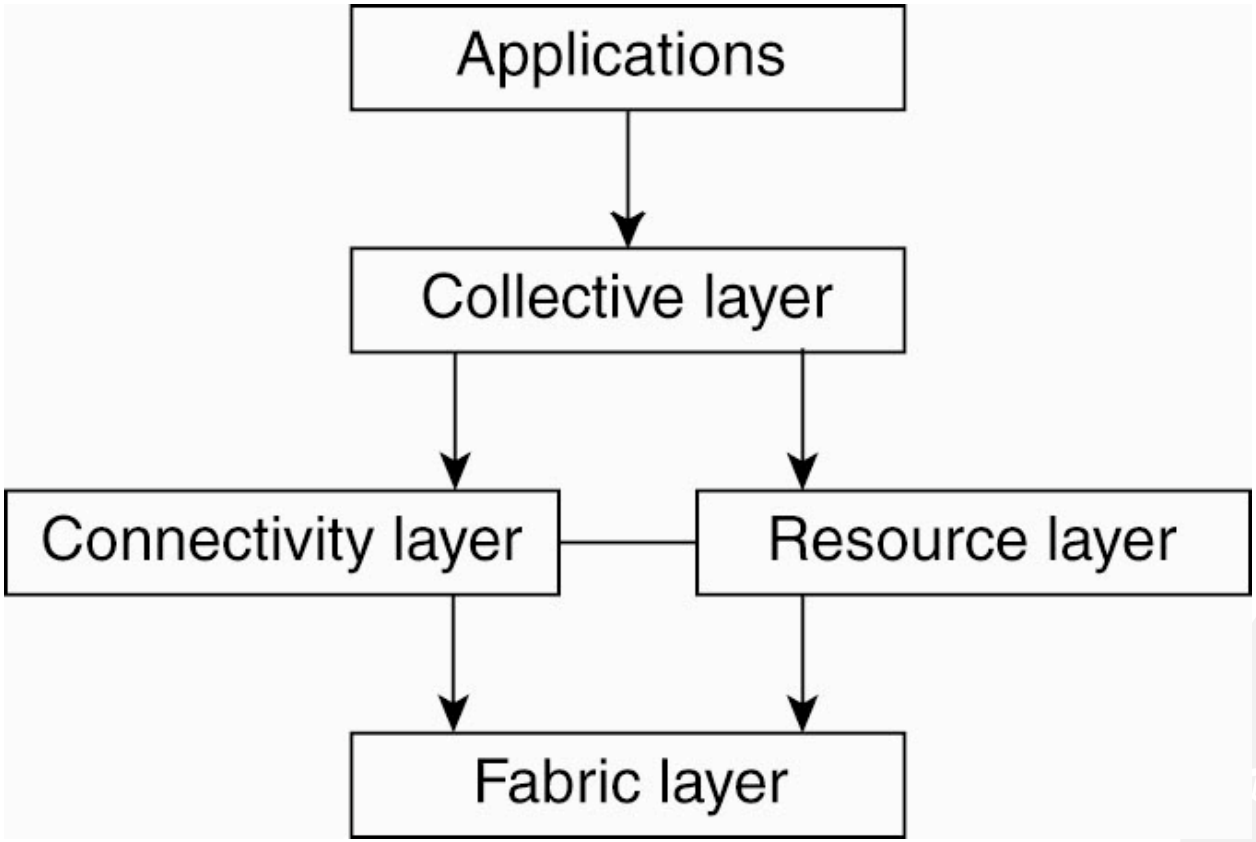
resource layer — management of single resources—e.g., access control

connectivity layer — communication protocols for grid transactions spanning over multiple resources, plus security protocols for authentication

collective layer — handling access to multiple resources—resource discovery, allocation, ...

application layer — applications operating in the virtual organisation

Architecture of a Grid Computing System II



[Tanenbaum and van Steen, 2007]

Architecture of a Grid Computing System III

Grid middleware layer

- the *core* of a grid middleware layer is represented by *connectivity*, *resource*, and *collective* layers
- altogether, they provide *uniform access* to otherwise dispersed *resources*

Distributed Information Systems

Origin

- many separate networked applications to be integrated
- structural problems of interoperability

Sorts

- several non-interoperating servers shared by a number of clients: distributed queries, distributed transactions

→ Transaction Processing Systems

- several sophisticated applications – not only databases, but also processing components – requiring to directly communicate with each other

→ Enterprise Application Integration (EAI)

Transaction Processing Systems I

Distributed transactions for distributed databases

- operations on databases are usually performed in terms of transactions
- when databases are distributed, transactions should be *distributed*
- *special primitives* from the distributed system or the runtime system
- *nested transactions* made of a number of subtransactions

ACID properties

atomic steps of a transaction occurs *invisibly* to the outside world

consistent the transaction does *not violate* system *invariants*

isolated *concurrent* transactions do *not interfere* with each other

durable once a transaction *commits*, its effects are *permanent*

Transaction Processing Systems II

Issue: durability of nested and sub-transactions

- a *whole* nested transaction should exhibit ACID properties
- so, if a subtransaction fails, all subtransactions till there should be *undone*, even though they already committed
- the effects of subtransactions could not be really durable if the whole transanction does not succeed

→ *durability* here refers to the *top-level* transaction

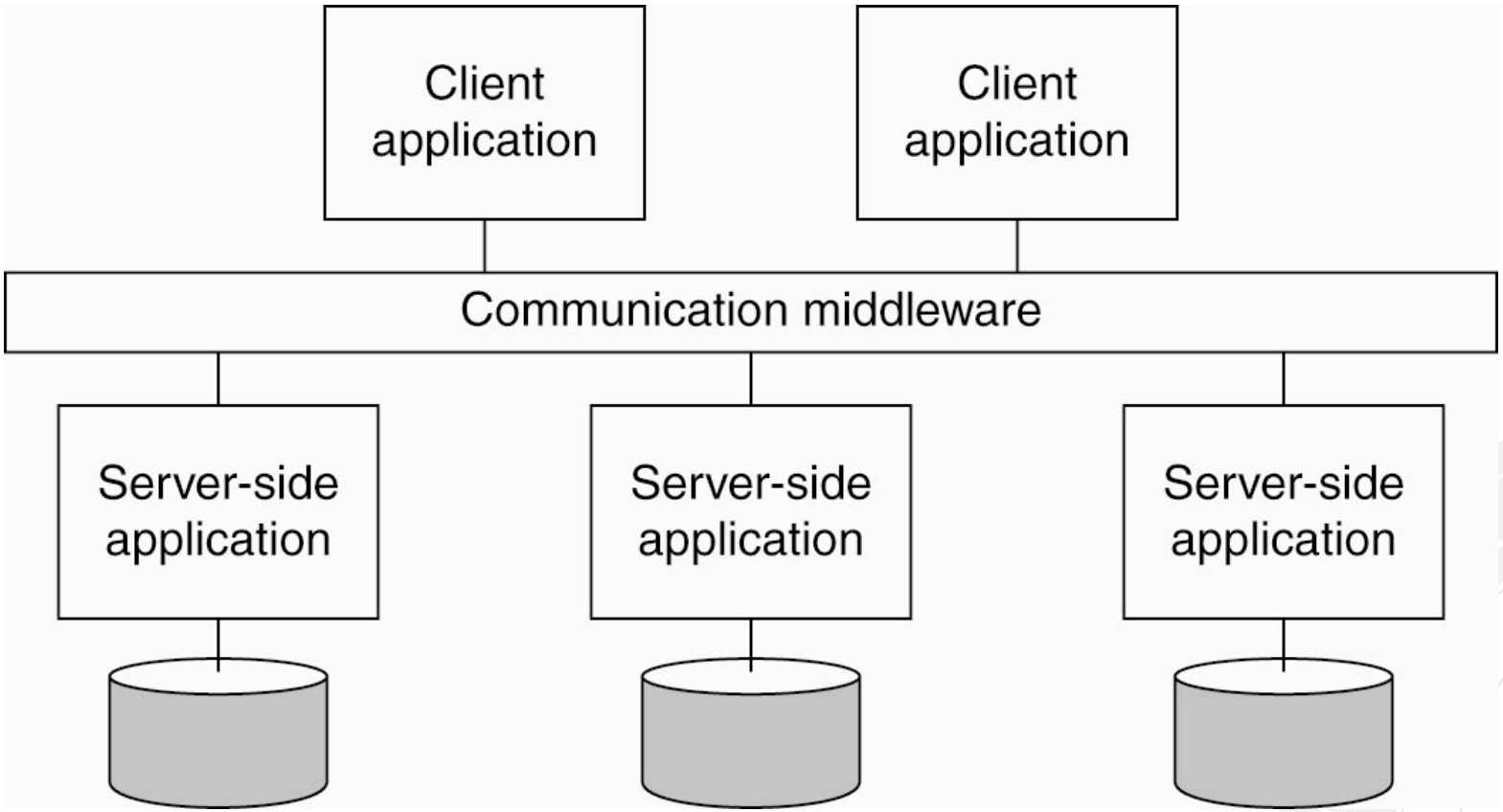
- *private copy* as a possible solution

Enterprise Application Integration

It is not only a matter of accessing distributed databases

- *integration* should happen at the application level, too
- beyond data integration, process integration
- application should *interact* and communicate *meaningfully* with each other

Middleware as a Communication Facilitator



[Tanenbaum and van Steen, 2007]

Distributed Systems with Instability

What happens when instability is the default condition?

- ... like, with mobile devices with batteries and sporadic network connection?
- ... like, in modern *distributed pervasive systems*?

Main features

- a distributed pervasive system is *part of* our *surroundings*
- a distributed pervasive system generally *lacks* of a *human* administrative *control*

Requirements for Pervasive Systems

Three requirements [Grimm et al., 2004]

- embrace contextual changes
- encourage ad hoc composition
- recognise sharing as the default

Remarks

- a device must be continually *aware* of the fact that its *environment may* change at any time
- many devices in pervasive system will be *used in different ways* by different users
- devices generally join the system in order to access (provide) information: *information* should then be easy to read, store, manage, and *share*

Home Systems

Systems built around home networks

- no way to ask people to act as a competent network / system administrator

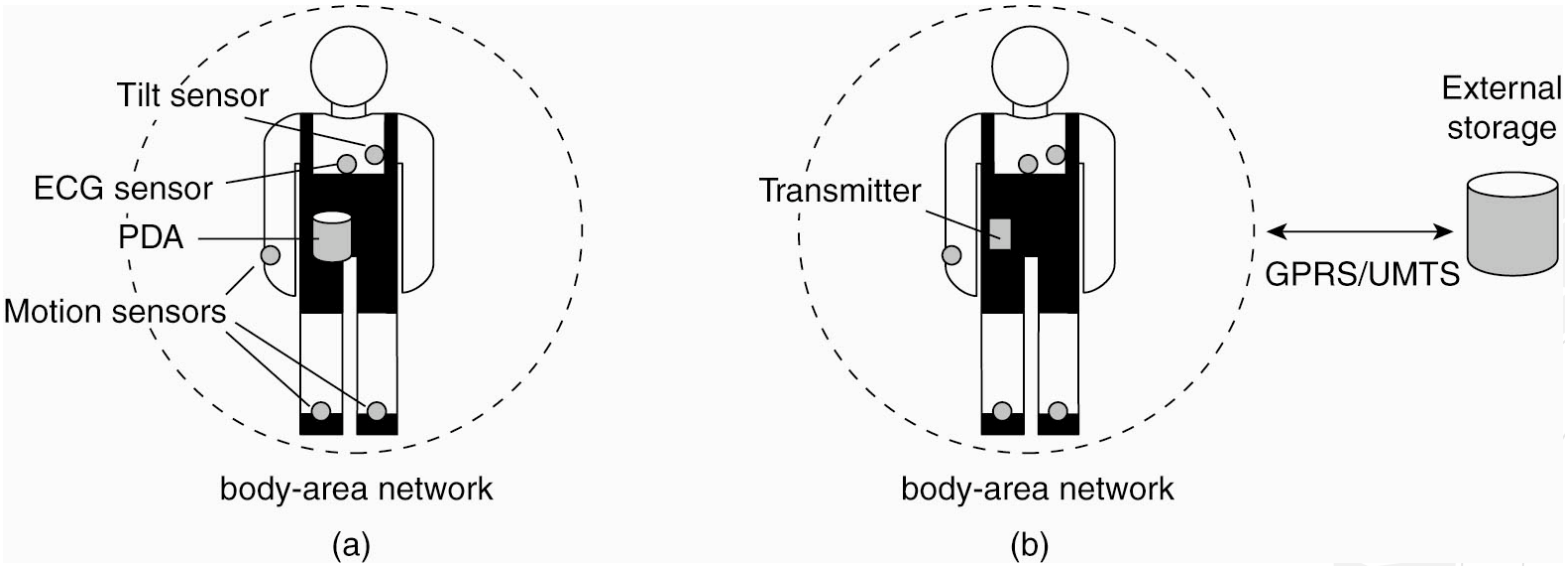
→ home systems should be self-configuring and self-maintaining in essence

Systems built around personal information

- *huge amount* of heterogeneous personal *information* to be managed,
- coming from *heterogeneous sources* from inside and outside the home system

Health Care Systems

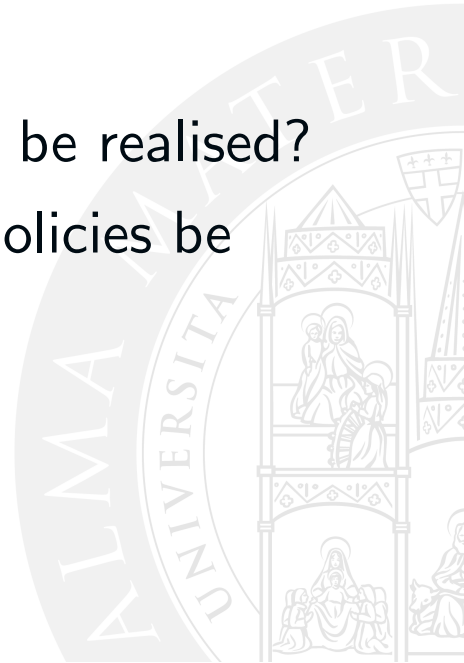
- Personal systems built around a Body Area Network
- Possibly, minimising impact on the person—like, preventing free motion



[Tanenbaum and van Steen, 2007]

Health Care Systems: Questions to be Addressed

- where and how should monitored *data* be *stored*?
- how can we prevent *loss* of crucial *data*?
- what infrastructure is needed to generate and propagate *alerts*?
- how can physicians provide online *feedback*?
- how can extreme *robustness* of the monitoring system be realised?
- what are the *security* issues and how can the proper policies be enforced?



Sensor Networks

An enabling technology for pervasive systems

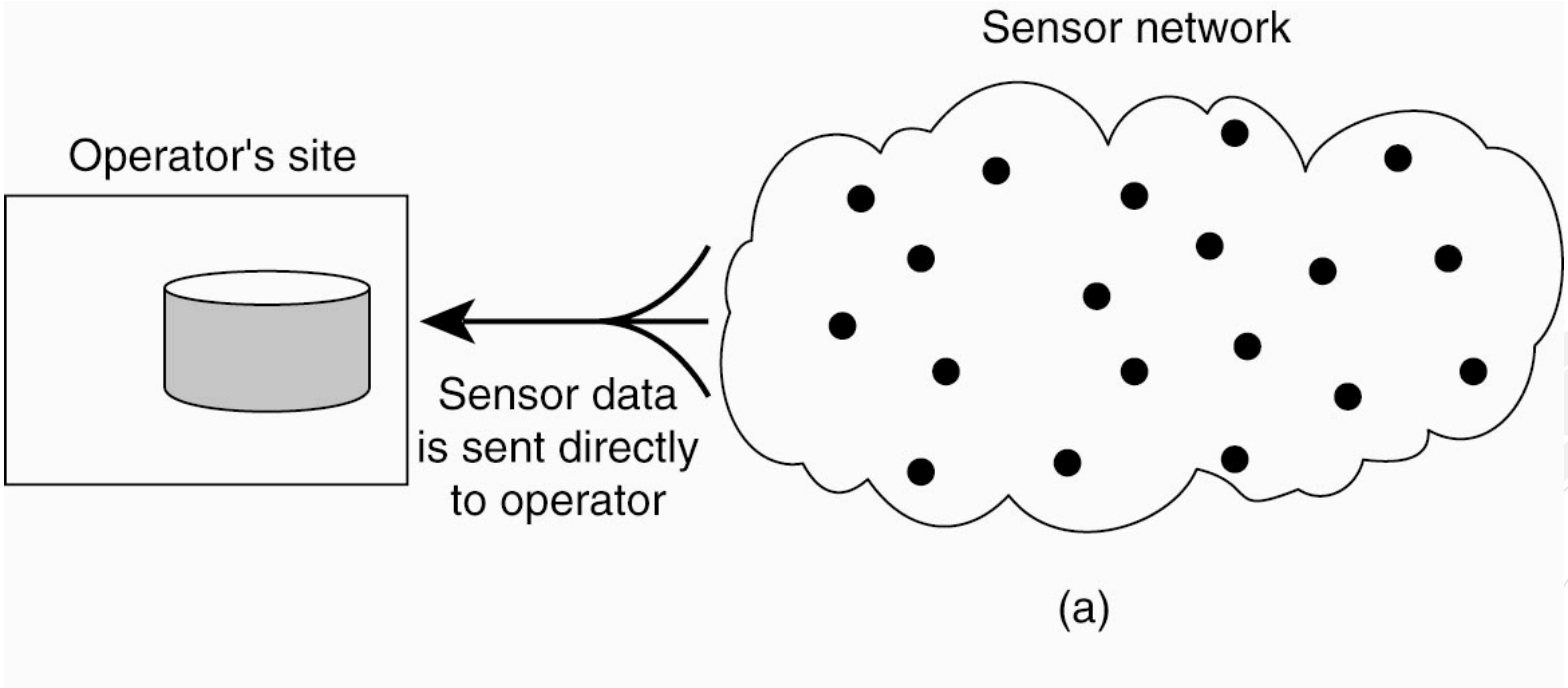
- clouds of *spatially-distributed sensors*—from tens to thousands of nodes with a sensing device
- acquiring, processing and transmitting *environmental information*

A possible view: distributed databases

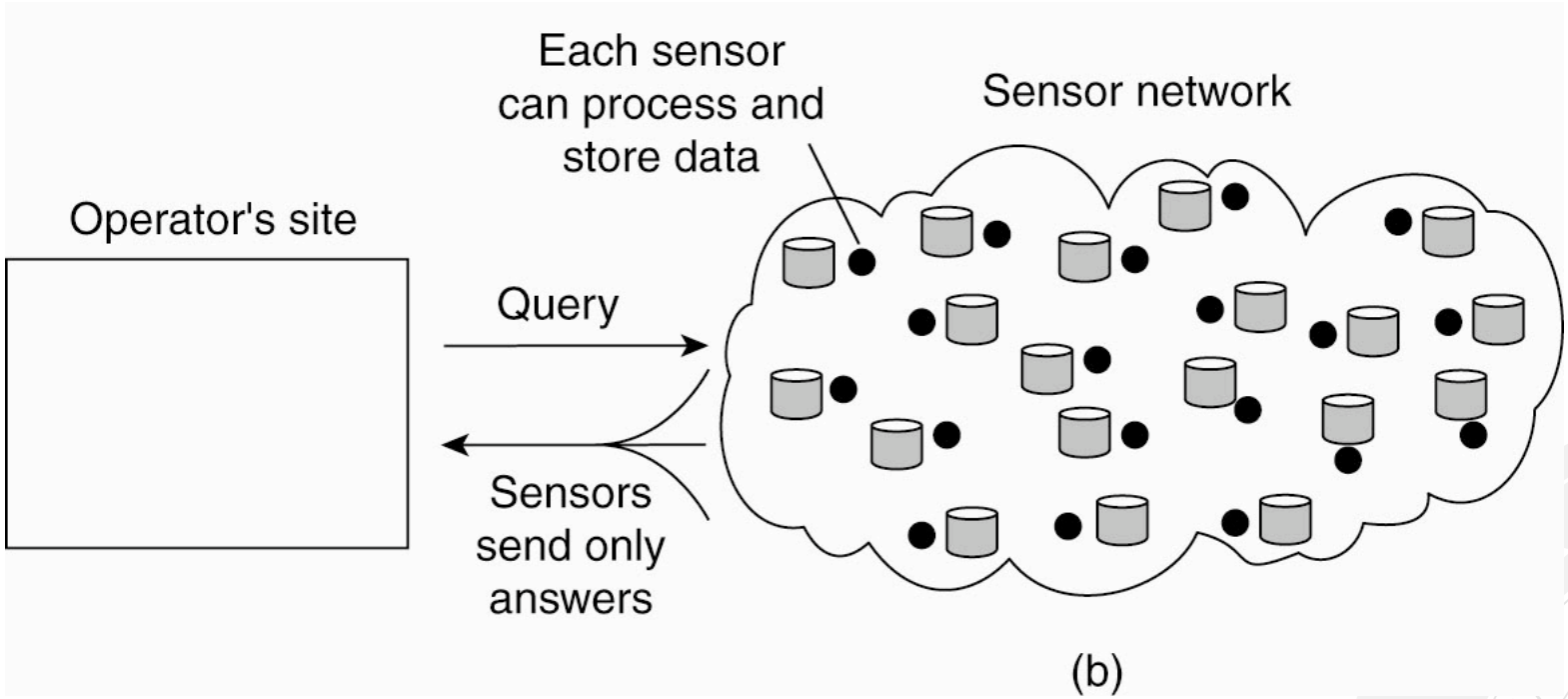
- distributed sources of information
- that can possibly be queried along time
- two possible extremes: either sensors just send information in without cooperating, or they do all the computation and return the results

↑ both solutions are bad ones, since they require either too much network consumption or too much node power consumption

Architecture of Sensor Networks: Two Extremes I



Architecture of Sensor Networks: Two Extremes II



[Tanenbaum and van Steen, 2007]

Sensor Networks: A Solution

In-network data-processing

- setting up a tree network among sensors
- passing queries through the sensor tree
- aggregating the results at the different levels of the tree

Questions to be addressed

- how do we (dynamically) set up an efficient tree in a sensor network?
- how does aggregation of results take place? Can it be controlled?
- what happens when network links fail?

Summing Up I

Foundations

- computer science
- computation

Basic abstractions

- abstract machines
- computational process
- context
- computational system

Summing Up II

Basic definitions

- interacting system
- parallel computation & system
- concurrent computation & system
- distributed computation & system

Evolution of computational systems

- distributed computing systems
- distributed information systems
- distributed pervasive systems

References I

Brooks, R. A. (1991).
Intelligence without reason.
In Mylopoulos, J. and Reiter, R., editors, *12th International Joint Conference on Artificial Intelligence (IJCAI 1991)*, volume 1, pages 569–595, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Burks, A. W., Goldstine, H. H., and von Neumann, J. (1982).
Preliminary discussion of the logical design of an electronic computing instrument.
In Randell, B., editor, *The Origins of Digital Computers*, Texts and Monographs in Computer Science, pages 399–413. Springer Berlin Heidelberg, Berlin, Heidelberg.

Conery, J. S. (2010a).
Explorations in Computing: An Introduction to Computer Science.
Chapman & Hall/CRC Textbooks in Computing. CRC Press, 1st edition.

Conery, J. S. (2010b).
Ubiquity symposium 'What is computation?': Computation is symbol manipulation.
Ubiquity, 2010(November):4:1–4:7.

References II

Degano, P. and Montanari, U. (1987).
Concurrent histories: A basis for observing distributed systems.
Journal of Computer and System Sciences, 34(2-3):422–461.

Denning, P. J. (2008).
The computing field: Structure.
In Wah, B., editor, *Wiley Encyclopedia of Computer Science and Engineering*, pages 615–623. John Wiley & Sons, Inc., Hoboken, NJ, USA.

Denning, P. J. (2010).
Ubiquity symposium 'What is computation?': Opening statement.
Ubiquity, 2010(November):1:1–1:11.

Denning, P. J. (2011).
Ubiquity symposium 'What is computation?': What have we said about computation?
Ubiquity, 2011(April):1:1–1:7.

Foster, I., Kesselman, C., and Tuecke, S. (2001).
The anatomy of the grid: Enabling scalable virtual organizations.
International Journal of High Performance Computing Applications, 15(3):200–222.

References III

Frailey, D. J. (2010).
Ubiquity symposium 'What is computation?': Computation is process.
Ubiquity, 2010(November):5:1–5:6.

Freeman, P. A. (2011).
Ubiquity symposium 'What is computation?': Is the symposium question harmful?: Consideration of the question 'What is computation?' considered harmful.
Ubiquity, 2011(March):1:1–1:4.

Goldin, D. Q., Smolka, S. A., and Wegner, P., editors (2006).
Interactive Computation: The New Paradigm.
Springer Berlin Heidelberg.

Grimm, R., Davis, J., Lemar, E., Macbeth, A., Swanson, S., Anderson, T., Bershad, B., Borriello, G., Gribble, S., and Wetherall, D. (2004).
System support for pervasive applications.
ACM Transactions on Computer Systems, 22(4):421–486.

Kshemkalyani, A. D. and Singhal, M. (2011).
Distributed Computing: Principles, Algorithms, and Systems.
Cambridge University Press.

References IV

Lamport, L. and Lynch, N. (1990).
Distributed computing: Models and methods.
In van Leeuwen, J., editor, *Handbook of Theoretical Computer Science, Volume B – Formal Models and Semantics*, chapter 18, pages 1157–1199. Elsevier.

Milner, R. (1993).
Elements of interaction: Turing award lecture.
Communications of the ACM, 36(1):78–89.

Newell, A., Perlis, A. J., and Simon, H. A. (1967).
Computer science.
Science, 157(3795):1373–1374.

Shonkwiler, R. W. and Lefton, L. (2006).
An Introduction to Parallel and Vector Scientific Computation.
Cambridge University Press.

Suchman, L. A. (1987).
Plans and Situated Actions: The Problem of Human-Machine Communication.
Cambridge University Press, New York, NYU, USA.

References V

Tanenbaum, A. S. and van Steen, M. (2007).
Distributed Systems. Principles and Paradigms.
Pearson Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition.

Turing, A. M. (1937).
On computable numbers, with an application to the entscheidungsproblem.
Proceedings of the London Mathematical Society, s2-42(1):230–265.

Wegner, P. (1997).
Why interaction is more powerful than algorithms.
Communications of the ACM, 40(5):80–91.

Wegner, P. and Goldin, D. (1999).
Mathematical models of interactive computing.
Technical report, Brown University, Providence, RI, USA.

Computing & Computer Science: Foundation, Evolution & Perspectives

Digital Culture

Andrea Omicini
andrea.omicini@unibo.it

Master in Business Management / Bologna Business School

Academic Year 2017/2018