

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



- 1 Computer Science
- 2 Computation
- 3 Computational Systems



Next in Line...

- 1 Computer Science
- 2 Computation
- 3 Computational Systems



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...

- 1 Computer Science
- 2 Computation**
- 3 Computational Systems



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,

- ❶ *A series of actions or steps taken in order to achieve a particular end. . .*
 - ❶ *A natural series of changes. . .*
 - ❷ *A systematic series of mechanized or chemical operations that are performed in order to produce something. . .*
 - ❸ *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)*

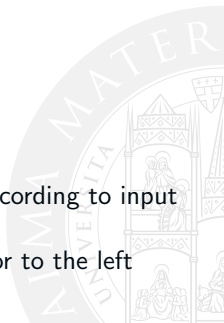
A computing machine is a different sort of machine. . .

- 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*

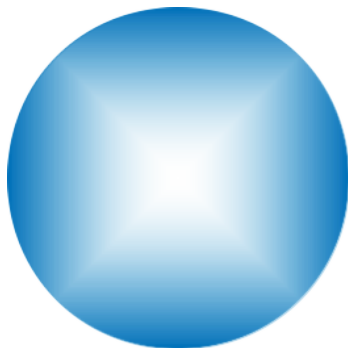
Computational Process I

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



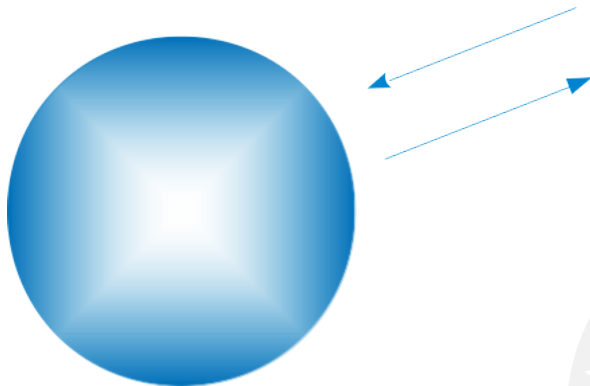
Computational Process II



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



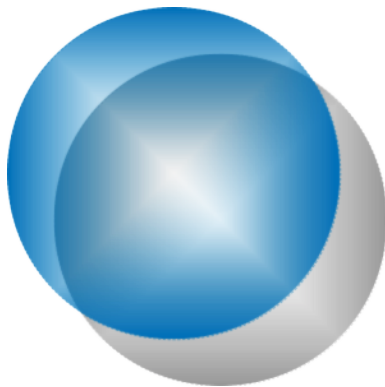
Computational Process III



Computational process with *input* and *output*



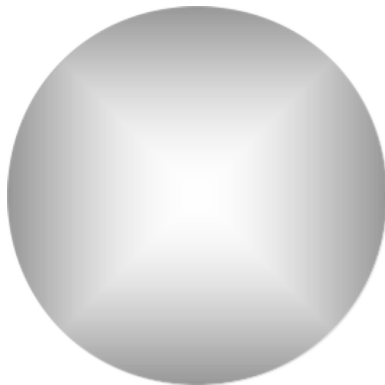
Computational Process IV



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

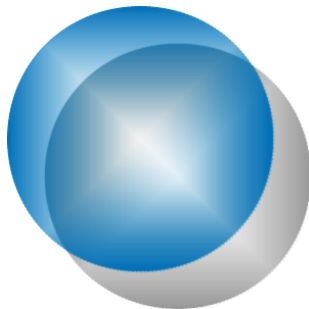
- **timed** computation, whenever the *time* of the computational machine is relevant / essential for the computing process
- **spatial** computation, whenever the *spatial features* of the computational machine are relevant / essential for the computing process
- more generally, **situated** computation [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...

- 1 Computer Science
- 2 Computation
- 3 Computational Systems**



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

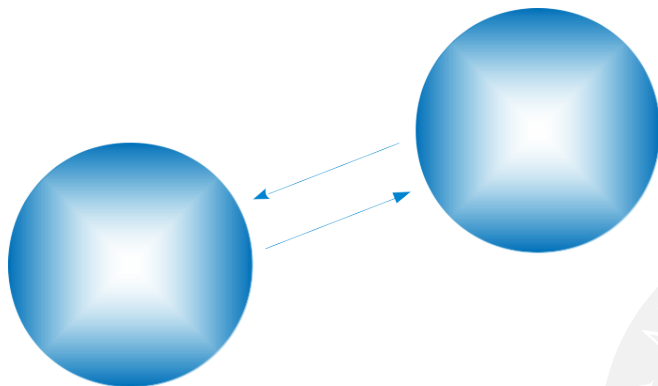
Computational system

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*



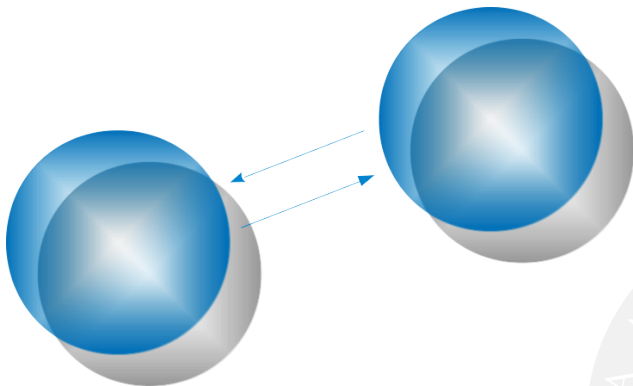
What is Context for a Computational System? I

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?



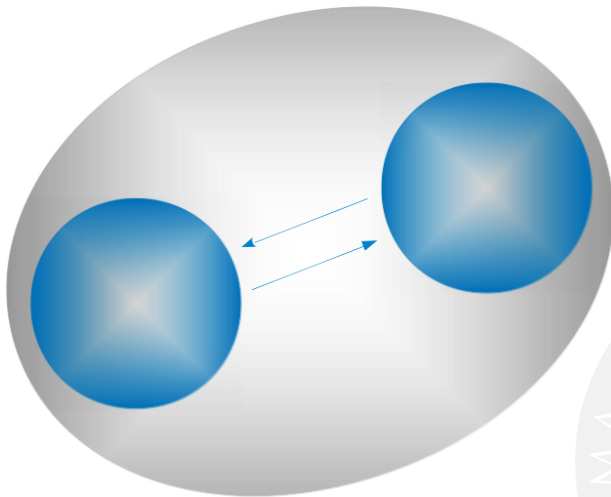
What is Context for a Computational System? II



A different context for each process



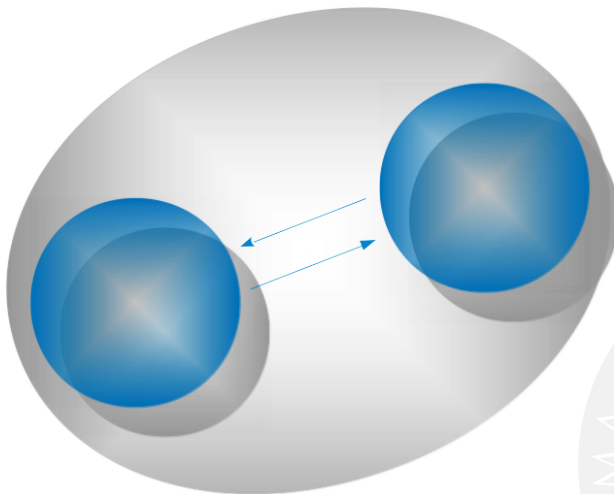
What is Context for a Computational System? III



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 acceptance

[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
- *spatial* context defines both



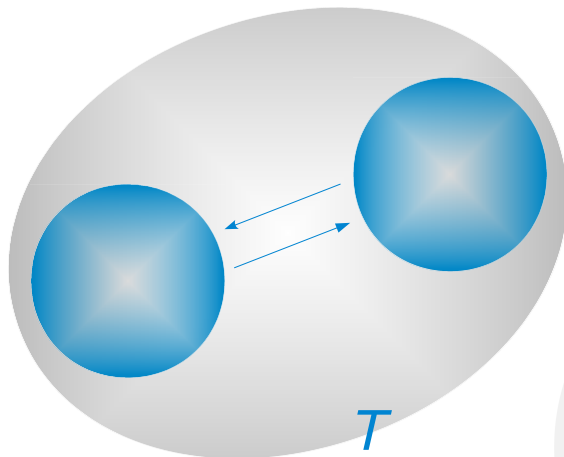
Definitions I

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

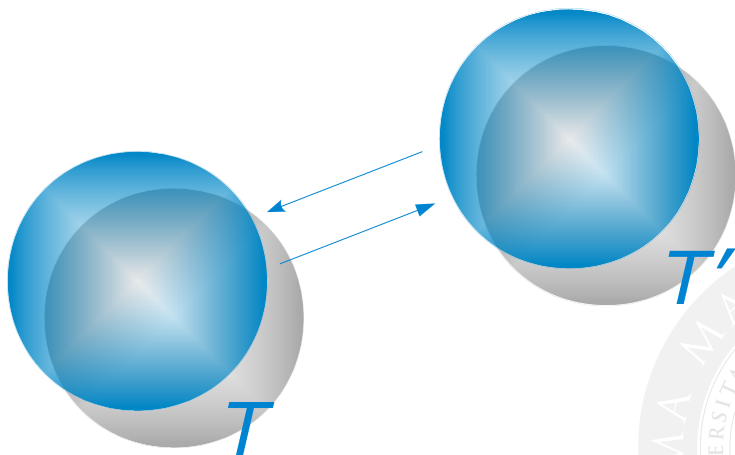
Definitions III

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

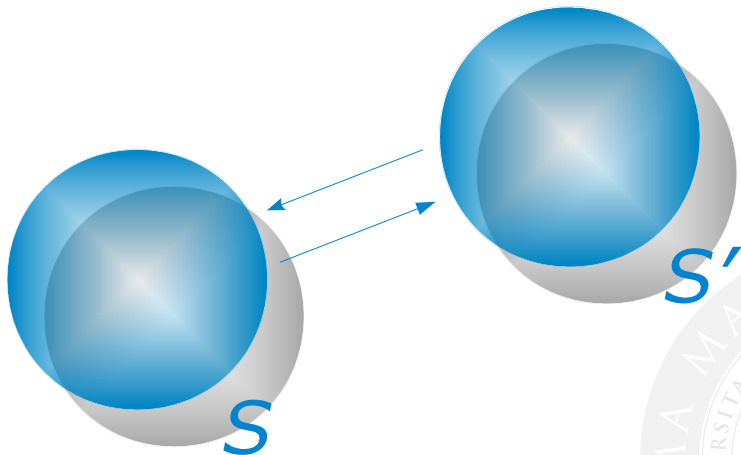
Definitions V

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

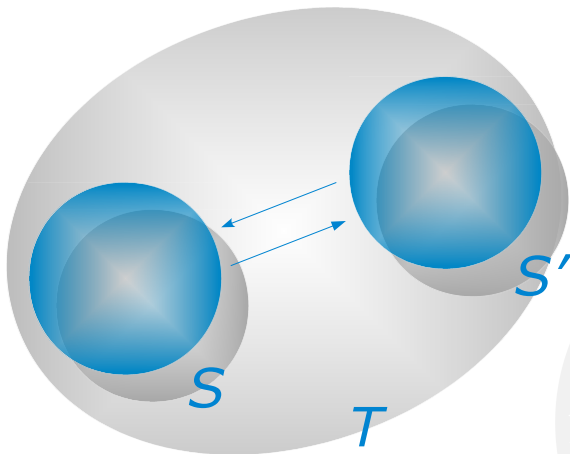


Definitions VI



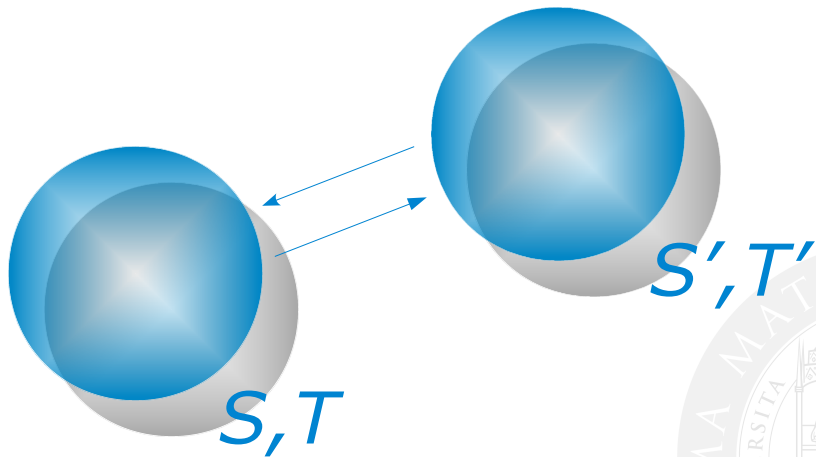
Distributed computing: different spatial contexts $S \neq S'$ for different processes

Spatial vs. Temporal Contexts I



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

Spatial vs. Temporal Contexts II



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

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



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.



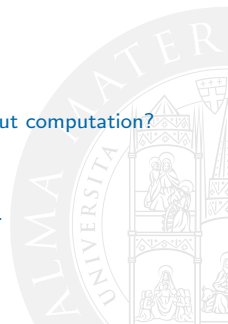
Denning, P. J. (2010).
Ubiquity symposium 'What is computation?': Opening statement.
Ubiquity, 2010(November):1:1–1:11.



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



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



References III



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.



Kshemkalyani, A. D. and Singhal, M. (2011).

Distributed Computing: Principles, Algorithms, and Systems.
Cambridge University Press.



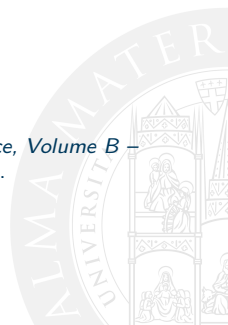
Lamport, L. and Lynch, N. (1990).

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.



References IV



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.



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.



References V

-  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

