Chapter 1
It's About Time

Imagine a world without time...

Can't do it? The egregious difficulty you have certainly experienced with such a Herculean stretch of imagination shows that the notion of time is so deeply entrenched in our mental models of the world that completely eliminating it is hardly possible.

Time plays a central role in everyday life, where thoughts, actions, and statements often include temporal references: "I'm going on vacation tomorrow", "I have to finish this job by the end of the week", "John is always late". Time is also a favorite object of philosophical inquiry, and a subject of religious speculation. The very notion of time pervades science and engineering.

Physics has dated the beginning of our universe to time $\mathscr{T}_0$ – approximately 13.7 billion years ago. Philosophy and religion, as well as physics itself, speculate about the state of affairs *before* $\mathscr{T}_0$. Past and future seem perfectly well defined notions at the level of intuition, but, whereas mathematics can treat the past as "future with reversed sign", physics establishes – with the second principle of thermodynamics – that the arrow of time does not go backward.

Engineering, concerned with the invention of systems that operate in the physical world and have some purpose, has to deal with several different notions of time: it affects human life, hopefully improving it; it applies knowledge about the laws of physics; it develops and analyzes mathematical models of the world; and, at least indirectly, it refers to speculations offered by philosophers, since philosophy is rooted in, is used to analyze, and in one way or another impacts our lives. For example, when engineering a subway system, the time saved by travelers using the transportation is a major driver in determining the routes and the frequency and speeds of trains. In addition, braking and acceleration times impact the passengers' safety and comfort, and are to be included in mathematical models of the cars.

Computer science has rapidly become pervasive in engineering and, as a consequence, in everyday life. Cars, for example, embed many electronic components, and also systems in many other different domains – banking, medical care,

C.A. Furia et al., *Modeling Time in Computing*, Monographs in Theoretical Computer Science. An EATCS Series, DOI 10.1007/978-3-642-32332-4_1,

transportation, etc. – depend, directly or indirectly, on the assistance of computing devices.

The proliferation of computing devices in the modern world is evident everywhere. What is, instead, less apparent is how the relatively short history of computing has produced novel problems and challenges in dealing with time, and solutions to them when designing systems with computational components.

This book has precisely the goal of analyzing this situation: on the one hand, computers are devices subject to physical laws like every other physical object; therefore, they can be modeled in terms of the motion of electrons in semiconductors and electromagnetic waves. On the other hand, computers work in widely diverse application domains, which implies dramatically different notions of time and its flow. For instance, users of automated teller machines certainly do not need a model of the electrons that flow in the circuits controlling the withdrawal of money; on the contrary, the users are concerned with the machine's responsiveness and expect to be able to receive money within a few seconds. Similar dualities occur in many other situations where people interact with computer-controlled devices.

The fundamental conceptual tool for coping with heterogeneous concerns in complex situations is *abstraction*, which consists of focusing on what is relevant in a certain context and for a certain purpose, while neglecting irrelevant details. Abstraction pervades computer science, which often has to deal with interacting multifarious domains as suggested by the examples above. Unsurprisingly, the models applied in computer science are often more diverse and heterogeneous than those in other sectors of engineering such as electrical or mechanical engineering, whose application domains are fairly established and well understood.

Abstraction of time is a special, and crucial, case: the roles and perceptions of time are heterogeneous, spanning very different domains – sometimes including psychological aspects ("happy times flow faster") – and hence models of time follow a great variety of approaches and have spawned diverse notations and formalisms.

At one extreme, given that computers are physical objects, we could model and analyze their behavior according to the physical laws of electromagnetism, which describe the flow of electrons through semiconductors or even the evolution of their quantum states.

At the other extreme, the theory of computation is fundamentally based on drastic abstractions of time, almost to the point of removing it completely from the models: in many traditional applications, only the results of a computational process matter, not so much how long it takes to obtain them. This was true with the slow batch computer systems of the past, when users input a collection of punched cards and came back after 1 day to pick up the printout of the results; but it also happens with the fast interactive computers of the present, when users perceive only the overall responsiveness of the system, and the time of each individual operation is negligible. In these scenarios, computational processes are abstracted as *functions* from input to output data.

Between these two extremes there is a continuum of abstractions and models, based on application environments, design goals, and preferences of the designers. Let us sketch a few examples, developed in greater detail throughout the book.

When moving from the point of view of electronic circuit design to that of hardware architecture, we apply *discretization*, namely the change from continuous to discrete domains. This applies both to time and to other domains used in the formal models. Discretization can be seen as a tool for mathematical analysis via numerical computation, which has burgeoned also in domains where time and dynamics are not primary concerns, for instance in the *finite element methods* for static analysis of structures.

Computational complexity theory defined another major historical approach to modeling time in computing. In some sense, computational complexity fills the "abstraction gap" of purely functional models, as it describes *how long* computations take, independently of what output they produce. Take, for instance, the problem of sorting a sequence of elements. First, we can describe and implement a few algorithms that obtain the desired result. Then, we classify the algorithms according to their complexity, preferring the most efficient ones, which require, say, a time proportional to $n \cdot \log n$ for every sequence with n elements. The computational complexity abstraction of time sharply departs from the traditional approaches in other fields of engineering, where system behavior is modeled by the evolution of *state as a function of time*. For example, the laws of mechanics describe the position and velocity of masses as functions of time, from which one can compute the time and space required by, say, a car to reach a full stop from a given initial speed.

However, the traditional view of computation as a sequential process that starts from some initial state, reads some input, and produces an output after some time is inadequate to model systems where computational elements work in collaboration with modules of different kinds. This is the case with so-called "reactive systems", which are often *embedded*. Reactive systems include computing devices as parts of a more complex system where different processes, with different dynamics, interact and *coordinate* with one another towards a common goal, or *compete* to access limited shared resources. Also, when the computations must obey *quantitative timing constraints* (e.g., "the shared resource cannot be occupied for longer than 100 seconds", "as a consequence of an alarm the system must be shut down within ten seconds"), the systems are called "real time".

The structure of reactive systems can be highly complex and they may include heterogeneous components that require diverse mathematical models. Often, the external *environment*, whose behavior is only partially controllable or observable, plays a prominent role in interacting with the other system components. The environment often includes users and actors – human or otherwise. For example, a car is a complex system made of interacting mechanical and electronic components, which interacts with a much larger and complex environment consisting of other cars, drivers, pedestrians, roads, and so on. Modeling, analyzing, and designing such systems requires the ability to formalize quite different features and their mutual interactions.

If we focus on time modeling, we notice how many different notions of time belong to different levels of abstraction. In the example of cars in traffic, there are, among other notions of time, those of revolutions per minute of the engines,

processor clocks in the electronic embedded components, reaction times of drivers, schedules of traffic lights, and so on. Such notions of time have quite different features and therefore require different mathematical models: the microseconds of electronic signals; the hours needed to go from city to city; the precisely determined time necessary to reach a full stop; the uncertain reaction time of average drivers from the instant an obstacle appears on the road to when the brake pedal is pushed. All these "times" belong to the same big picture; competent designers must be able to analyze their dynamics in isolation whenever possible, but also be able to understand their interactions when relevant – for instance, when documenting the behavior of brakes from the user's perspective.

Heterogeneity, however, is not always an issue: a special class of systems consists of collections of homogeneous components that cooperate towards a common goal. This is the case, for example, with multiple identical pistons and cylinders in a car, which together have more power than a single cylinder could have, or of the parallel processors in a multi-core machine. With homogeneous components, coordination and synchronization become the main modeling and design concerns.

In response to the advent and rapid ongoing evolution of heterogeneous reactive systems, the scientific community has developed a rich collection of formalisms, notations, and techniques to deal with the various aspects of timing analysis. The introduction and evolution of the modeling notations has inevitably often been haphazard and demand-driven, corresponding to the evolving needs of applications. As a result, publications describing specific approaches, methods, and tools abound, but there is a lack of comprehensive systematic analyses that investigate general issues and survey the peculiarities of the different contributions.

Filling this void is the main goal of this book, which aims at fostering the critical thinking of readers towards:

- Understanding the subtleties of system dynamics when analyzing problems and investigating possible solutions (we will see that time is often "hidden" in models that do not feature it explicitly);
- Evaluating and comparing models and approaches and selecting the most appropriate ones for the specific needs (we will see that, unlike in other fields of engineering, the "best" formalisms are not always evident; on the contrary, tailoring and integrating existing solutions may be necessary in some new cases).

To achieve these goals, the book develops in two main directions. It presents some fundamental categories useful for comparing and evaluating modeling notations encapsulating time. These categories include issues such as whether time is modeled as a discrete or a dense domain. The book's other, orthogonal, direction is historical, which starts with a review of the traditional time models in science and engineering in general, and in computer science in particular. The presentation continues with more recent models that address specifically the situation of complex systems where computing devices interact with subsystems of other types. In this respect, it is important to emphasize how an interdisciplinary approach is becoming more and more relevant in modern system design: with the exception of very few highly specialized fields, it is essential that software designers understand the

application domain and, conversely, domain engineers have a working knowledge of the computing subsystem's behavior and of its interactions. The same interdisciplinary approach may be relevant also for the general public, beyond the technicians and engineers, since, as we emphasized before, human-computer interaction is a primary attribute of many complex systems.

Within this global picture, time plays a fundamental role, on the one hand being the unifying variable that spans the life of the whole universe, on the other hand showing itself in so many different ways and forms to the various actors of the universe's life, from subnuclear particles that exist for a few nanoseconds to stars that "die" billions of years after they "have been born", from the pace of a human heart to the time needed to obtain a university degree.

In correspondence with the above directions, the book is structured into three introductory chapters and two parts, and concluded by a short epilogue. After this introduction, Chap. 2 presents the notions of formalism and model in general terms, and some of their fundamental classification criteria; it also briefly discusses the fundamentals of propositional and predicate logic, which should help make the rest of the book self-contained for a reasonably large readership.

Chapter 3 is a cornerstone of the whole book, as it introduces a taxonomy of essential issues of modeling time in diverse systems. The presentation of the numerous formalisms in the rest of the book recurrently refers to these "dimensions" to compare and contrast different models on a common ground.

Part I contains a concise summary of the models of time that are traditional in engineering and the natural sciences, including traditional computer science. It is meant to provide heterogeneous readers with a homogeneous background.

Part II covers advanced and specialized formalisms specifically developed to support time modeling in heterogeneous software-intensive systems. The aim of Part II is not to offer an exhaustive list of the innumerable contributions available in the literature; this would be a Herculean task, but also probably of little value. On the contrary, the presentation privileges depth over exhaustiveness, and focuses on significant semantic subtleties of a few important formalisms and critical issues, rather than cataloging every minimal variation of the basic approaches. Readers interested in additional details will still find detailed, commented bibliographic references at the end of each chapter. We hope that this presentation style will help readers extend the analysis to other paradigms or approaches not included in the main text. Chapters 7–9 discuss three main and complementary families of formalisms: those based on finite state machines; Petri nets; and those extending mathematical logic. Chapter 10 is about process algebras – widely used to model concurrency, but less prominently so in timing analysis. Chapter 11 presents "dual-language approaches" which combine two notations with different characteristics to model and verify complex systems (model checking frameworks are the most popular applications of dual-language approaches).

Chapter 12 concludes the book with summarizing remarks and hints towards future developments and challenges.

The book's content focuses on the way formalisms can be used to model system behavior and properties and on their expressive power – also in the informal sense

of naturalness and ease of use. Analysis and verification techniques and tools for the various formalisms have, in contrast, a more limited coverage, as the book is not meant to focus on verification techniques. Nevertheless, every chapter in Part II includes a section that mentions analysis and verification techniques and tools based on the notations introduced in the chapter.

1.1 Bibliographic Remarks

While it is arguable that *Homo sapiens* began thinking about time shortly after the development of natural language, ancient philosophers were the first whose observations have been recorded and preserved to this day, and have influenced the evolution of science and engineering. The rest of this section gives a very sketchy outline of some of these ideas [8, 10].

The Greek pre-Socratic (and pre-sophistic) naturalist philosophers of the fifth and fourth centuries B.C. suggested informal models of universal time. Some of them, most notably Heraclitus and his followers, predicated a notion of time that is "monotonic" (using modern terminology) in that it never repeats itself; others, most notably those from Parmenides's school, considered time an illusion devoid of physical reality. Among Parmenides's disciples, Zeno of Elea has become famous for his paradoxes on the advancement of time; some critical behaviors in the formal analysis of systems have been named after him (see Sect. 3.6). Many philosophers following the Greek naturalists have adopted, and refined, either Parmenides's or Heraclitus's ideas about time; some thinkers, such as Vico in the seventeenth century or Nietzsche in the nineteenth, have developed an intermediate view where time undergoes real progress but periodically repeats itself.

Kant's gnoseology describes time as an a priori concept, ingrained in the human mind and hence usable as a universal reference in describing the physical world. This view bolstered the development of classical Newtonian mechanics; when Einstein generalized Newton's models with his Theory of Relativity, he also had to perfect its philosophical underpinnings to account for the fact that different observers measure time differently according to their relative motion. Contemporary physics, with its experiments and speculations, keeps on questioning the traditional views of time and introduces new, original explanatory models.

Literature developed around original notions of time is also abundant, as it includes a large part of science-fiction books and movies that entertain the possibility of time-travel; since it is impossible to cite even a fraction of these many books, let us just mention the irresistible description of the problems of grammar related to time travel in Douglas Adams's "The Restaurant at the End of the Universe" [1]. The notion of time in science has also inspired some major literary masterpieces, such as some of Borges's short stories [3], and several of Calvino's novels [4, 5].

The rest of this book offers many specific technical references on time. More general examples of recent papers that discuss some "philosophical" aspects of

time with technical rigor include Alur and Henzinger's well-known surveys [2, 7] (the second survey, [7], shares the title with this chapter), Koymans [9], and Schreiber [11]. Finally, the same basic motivations that spawned our survey paper [6] also guided us in developing this book.

References

1. Adams, D.: The Restaurant at the End of the Universe. Pan Macmillan, London (1980)
2. Alur, R., Henzinger, T.A.: Logics and models of real time: a survey. In: Real Time: Theory in Practice. Lecture Notes in Computer Science, vol. 600, pp. 74–106. Springer, Berlin/New york (1992)
3. Borges, J.L.: Collected Fictions. Penguin, New York (1969)
4. Calvino, I.: Cosmicomics. Harcourt Brace, New York (1968). Original Italian title: *Le cosmicomiche*
5. Calvino, I.: t Zero. Harcourt Brace, New York (1969). Original Italian title: *Ti con zero*
6. Furia, C.A., Mandrioli, D., Morzenti, A., Rossi, M.: Modeling time in computing: a taxonomy and a comparative survey. ACM Comput. Surv. **42**(2), 1–59 (2010). Article 6
7. Henzinger, T.A.: It's about time: real-time logics reviewed. In: Sangiorgi, D., de Simone, R. (eds.) Proceedings of the 9th International Conference on Concurrency Theory (CONCUR'98). Lecture Notes in Computer Science, vol. 1466, pp. 439–454. Springer, Berlin/New York (1998)
8. Hetherington, S. (ed.): Epistemology: The Key Thinkers. Continuum, London/New York (2012)
9. Koymans, R.: (Real) time: a philosophical perspective. In: de Bakker, J.W., Huizing, C., de Roever, W.P., Rozenberg, G. (eds.) Proceedings of the REX Workshop: "Real-Time: Theory in Practice". Lecture Notes in Computer Science, vol. 600, pp. 353–370. Springer, Berlin/New York (1992)
10. Russell, B.: A History of Western Philosophy. Simon and Schuster, New York (1967)
11. Schreiber, F.A.: Is time a real time? An overview of time ontology in informatics. In: Halang, W.A., Stoyenko, A.D. (eds.) Real Time Computing. NATO ASI, vol. F 127, pp. 283–307. Springer, Berlin/New York (1994)

http://www.springer.com/978-3-642-32331-7

Modeling Time in Computing

Furia, C.A.; Mandrioli, D.; Morzenti, A.; Rossi, M.

2012, XVI, 424 p., Hardcover
ISBN: 978-3-642-32331-7