Information and Computation 241 (2015) 277–301

Contents lists available at ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

Computability and realizability for interactive computations

Manfred Broy

Institut für Informatik, Technische Universität München, 80290 München, Germany

ARTICLE INFO

Article history:
Received 13 February 2009
Received in revised form 24 January 2013
Available online 6 January 2015

Keywords:
Computability
Interaction
Realizability

ABSTRACT

This paper deals with computability of interactive computations. It aims at the characterization and analysis of a general concept of interactive computation as a basis for the extension and generalization of the notion of computability to interactive computations. We extend the notion of computability to interactive computations. Instead of partial functions on natural numbers or on finite strings we work with functions and relations on infinite input and output streams.
As part of the computability of such functions and relations on streams we treat the following aspects of interactive computations:

- causality between input and output streams
- realizability of single output streams for given input streams
- the role of non-realizable output
- relating non-realizable behaviors to state machines
- the concept of interactive computation and computability for interactive systems
- the role of time in computability.

Finally, we relate our results to established notions of computability.

1. Motivation

In contrast to the early days of computing with their pioneers Turing, Zuse, and von Neumann, where computers and programs were used to compute *output values* from *input values* by sequential algorithms implementing *partial functions*, today computers and their computations are fundamentally interactive. Computations interact with each other when executed on processors in a multiplexing mode. Computations interact with their environment via communication devices connecting them to sensors and actuators, to neighbor systems, and to peripheral devices realizing human machine interfaces for their human users. They interact mutually when executed on sets of processors connected by communication links. Interactive distributed computations are standard for information processing systems today.

Traditional computability theory deals with so-called sequential algorithms and sequential, non-interactive computations. The concept of a “non-interactive” computation is considered so fundamental that computing science has not even created a specific name for it – historically a computation was considered non-interactive per se. Computation was considered as a process that after having provided its input – be it by a set of parameters for a function or by some initial state – carries out its computation steps sequentially without any interaction with its environment until it eventually stops and delivers

E-mail address: broy@in.tum.de.
URL: http://www.broy.informatik.tu-muenchen.de.

http://dx.doi.org/10.1016/j.ic.2014.12.019
0890-5401/

its result. If it does not stop and carries on being engaged in a non-terminating computation it was called a *divergent* computation and considered useless, since it did not deliver any result in the end. Even the fact that it did not terminate could not be understood actually as a "delivered result" since non-termination cannot be recognized by finite observations. The property of non-termination was shown to be undecidable. Partial functions therefore proved to be the appropriate concept to model what non-interactive algorithms compute.

Church's thesis (see [10]) that μ-recursive functions (and equivalently Turing machines) capture the intuitive notion of computability for sequential, non-interactive algorithms is universally accepted. This acceptance is supported by the fact that μ-recursive computability is equivalent to other fundamental notions of computability such as Turing-computability or register-machine computability.

However, Church's thesis deals solely with non-interactive computations. For interactive (see [20,30]), concurrent (see [15]), distributed, real-time computations the notion of Turing-computability is not directly applicable and less well understood (see [16–18]). Concepts like infinite non-terminating computations and fairness bring in new questions (see [3]). It has been claimed several times that interactive, concurrent, distributed, real-time systems lead to computations that go beyond Church's thesis (see, for instance, [25] and also [24]). In fact, concurrency introduces non-determinism and asks for concepts like fairness, which leads to unbounded non-determinism, leaving the ground of classical computability. This can be justified by the postulate that "it is not constructively possible to choose from an infinite set one of its elements in a fair manner in bounded time" (see also [1] and [4]). This relates also to the well-known lemma (König's lemma) that in a tree that is only finitely branching and contains an infinite number of nodes there exists an infinite path. So if we assume that in each effective ("finite") step of computation only a finite number of choices can be done then to choose one element from an infinite set (which is the case under the condition of fairness) necessarily includes a non-terminating path. Since fairness, a concept often found in models of concurrent computations, requires excluding the infinite non-terminating branches, by fairness assumptions the ground of Turing computability is abandoned (for details see [9]).

In computability theory, a key notion is the concept of a *computation* problem given by a problem specification and moreover that of a *computable problem* characterizing a *computable function*. A computation problem is given by (the description of) a partial function that is to be computed. The set of all partial functions is seen as appropriate representation of the set of all computation problems. A problem is algorithmically solvable ("computable") if the partial function characterized by the problem is computable. A partial function is computable if there exists an algorithm (a Turing machine, a μ-recursive expression) that computes it. For interactive computations, however, a more sophisticated concept than partial functions is needed.

The Focus theory (see [5]) provides an abstract descriptive theory for engineering of distributed interactive systems supporting a modular, component-oriented system modeling and interface specification. Its key goal is interface abstraction that supports a descriptive approach to interactive systems. An interface specification describes a problem to be computed by a state machine. The theory is based on the notion of *causality* capturing the causal relationship between input and output in interactive computations, as well as *realizability*, at a first glance a less intuitive concept that captures the question whether there is an interactive strategy, for instance in terms of a state machine, to solve a given problem. Both causality and realizability are concepts that characterize the proper flow of messages of interactive systems in a time frame. The concept of a causal, realizable interactive behavior is initially introduced as representation of the interface behavior of interactive systems. Notions such as causality and realizability characterize properties that are of interest for the specification, the modeling and engineering of systems. In this paper, we apply these notions in a more theoretical way, targeting the issue of computability for interactive computation. We give a concise analysis of the idea of realizability. Furthermore, we relate this notion to classical concepts of computability.

Computability is a real world issue that can be captured by a formal theory. The question "*Can we construct real world machines that compute solutions to a given class of a problems*" gets more precise by defining which kind of machines and which kind of problems are considered. We restrict ourselves to digital devices with a pulse driven execution. As a consequence, we restrict our approach to discrete time. Notions of computability for systems working with continuous time and continuous functions as studied in control theory (see [22]) are not considered in the following.

When analyzing computability by formal concepts the choice of the models and abstractions is crucial. As soon as a specific formal model of computation is chosen – such as Turing machines, recursive functions, or register machines – a specific notion of formal computability is implicitly predetermined. Then a careful analysis is needed to show consequences and implications of the chosen formalism. Church's thesis addresses this issue: different models of computation have been chosen – however, according to Church's thesis there is a canonical idea of computability captured by the set of partial functions that are identified as effectively computable. As a result the notion of computability of a partial function is independent of the chosen formal concept of computation – as long as a sufficiently powerful model of computation is selected that captured the intuitive concept of mechanical evaluation.

The history of computability theory shows how much notions of computability depend on the chosen formal concepts and abstractions. For instance, if we restrict models of computing devices to finite state machines – all real world digital machines are finite state – we end up with a rather restricted notion of computability. Considering machines with infinite state spaces and unbounded states is an idealized but helpful view that leads to a more powerful concept of computation.

In this paper, we choose a straightforward notion of discrete time and interaction. The introduction of time into models of computation makes an essential difference to notions of computability without explicit notions of time. We use a simple notion of discrete time in the following where time is represented by an infinite chain of time intervals of equal length