On Interactive Data Structures

WALTER DOSCH

Institute of Software Technology and Programming Languages
University of Lübeck
Lübeck, Germany
http://www.isp.uni-luebeck.de

Summary

Modern computer systems are composed of software and hardware components of different kinds that communicate (a)synchronously by exchanging information along connecting channels. Over the years, the computing paradigm shifted from sequential systems to component-based distributed systems.

Characteristically, a sequential process describes a finite computation where input is provided in the initial state and output is generated in the final state. Unlike a sequential process, an interactive component shows a potentially infinite behaviour where partial input is received during an ongoing computation and incremental output is generated in reaction to the input received. As time progresses, an interactive component consumes a stream of input messages and produces a stream of output messages.

Software and hardware components often encapsulate a data structure as the component's internal state. In the algebraic approach, a data structure is described as a multi-sorted algebra composed from carrier sets, constants and operations. The algebraic approach supports well the specification of abstract data types and the paradigm of functional programming. The static description of data structures, however, does not cover the dynamic aspects of interactive components.

We present a formal method how to transform a functional data structure in a systematic way into an interactive component. In the first design step, we convert the functional interface of the data structure into an interaction interface. In the second design step, we derive the component's input/output behaviour as a relation between input histories and output histories. In the third design step, we implement the component's input/output behaviour as a state transition machine introducing states as history abstractions.

The formal method provides a safe roadmap from functional data structures to state-based implementations as interactive components. The approach contributes to a better understanding of functional, communication-oriented and state-based descriptions of data structures.