Chapter 18

The Formal Design Models of Tree Architectures and Behaviors

Yingxu Wang
University of Calgary, Canada

Xinming Tan
Wuhan University of Technology, China

ABSTRACT

Trees are one of the most fundamental and widely used non-linear hierarchical structures of linked nodes. A binary tree (B-Tree) is a typical balanced tree where the fan-out of each node is at most two known as the left and right children. This paper develops a comprehensive design pattern of formal trees using the B-Tree architecture. A rigorous denotational mathematics, Real-Time Process Algebra (RTPA), is adopted, which allows both architectural and behavioral models of B-Trees to be rigorously designed and implemented in a top-down approach. The architectural models of B-Trees are created using RTPA architectural modeling methodologies known as the Unified Data Models (UDMs). The physical model of B-Trees is implemented using the left and right child nodes dynamically created in memory. The behavioral models of B-Trees are specified and refined by a set of Unified Process Models (UPMs) in three categories namely the management operations, traversal operations, and node I/O operations. This work has been applied in a number of real-time and nonreal-time system designs such as a real-time operating system (RTOS+), a general system organization model, and the ADT library for an RTPA-based automatic code generator.

DOI: 10.4018/978-1-4666-2651-5.ch018
1. INTRODUCTION

Data structure modeling is a process to creatively extract and abstractly represent a real-world problem by data models based on constraints of given computing resources. Abstract Data Types (ADTs) are a set of highly generic and rigorously modeled data structures in type theory (Guttag, 1977; Broy et al., 1984; Cardelli & Wegner, 1985; Stubbs & Webre, 1985), which is an abstract logical model of a complex data structure with a set of predefined operations. A number of ADTs have been identified in computing and system modeling such as stack, queue, sequence, record, array, list, tree, file, and graph (Broy et al., 1984; Mitchell, 1990; McDermid, 1991; Wang, 2007; Wang, Ngolah, Tan, Tian, & Sheu, 2010). ADTs possess the following properties: (i) An extension of type constructions by integrating both data structures and functional behaviors; (ii) A hybrid data object modeling technique that encapsulates both user defined data structures and allowable operations on them; (iii) The interface and implementation of an ADT are separated where detailed implementation of the ADT is hidden to applications that invoke the ADT and its predefined operations.

A tree is a non-linear hierarchical structure of linked nodes (Stubbs & Webre, 1985; McDermid, 1991; Gosling, Bollella, Dibble, Furr, & Turnbull, 2002), which can be formally modeled by an ADT. Trees can be classified, according to their topology, into the categories of oriented trees (in which the finite nodes except the root node are partitioned into $k$, $k \geq 0$, disjoint sub-oriented trees), ordered trees (an oriented tree where the subtrees are ordered by their subscripts), binary trees (in which the finite nodes except the root are partitioned into $n$, $0 \leq k \leq 2$, disjoint sub-binary trees), $n$-nary trees (in which the finite nodes except the root are partitioned into $n$, $0 \leq k \leq n$, disjoint sub-$n$-nary trees), complete trees (in which the subnodes of all nodes except the leave nodes are complete) (Stubbs & Webre, 1985; Wiener & Pinson, 2000). Trees may also be classified according to their types of applications such as search trees (an ordered binary tree where for a certain node, all keys of nodes in the left and right subtrees are less or greater, respectively, than the key of the given node), parse trees (an oriented tree for syntax structures of compilers), organization trees (a complete $n$-nary tree for system organization and management), and decision trees (an ordered tree with nodes as value/alternatives and links as choices) (Stubbs & Webre, 1985; Wang, 2007; Wang & Ruhe, 2007). Various important data objects and system architectures can be modeled and implemented by trees such as a set of data with keys, a syntactic parse tree, and an expression tree (McDermid, 1991).

In order to formally model the tree ADT, an expressive denotational mathematics, Real-Time Process Algebra (RTPA) (Wang, 2002, 2007, 2008a, 2008b, 2008c, 2008d; Wang, Tan, & Ngolah, 2010), is adopted, which allows both architectural and behavioral models of lists to be rigorously designed and implemented in a top-down approach. According to the RTPA methodology for system modeling and refinement, a formal list can be rigorously modeled using two fundamental techniques known as the unified data model and the unified process model (Wang, 2007).

**Definition 1:** A **Unified Data Model** (UDM) is a generic architectural model for a software system as well as its internal control structures and its interfaces with hardware components, which can be rigorously modeled and refined as an $n$-tuple, i.e.:

$$UDM \triangleq \bigcup_{i=1}^{n} (S_i | \forall e \in S_i, p_i(e))$$

where $\bigcup_{i=1}^{n}$ is the big-R notation of RTPA that denotes a repetitive or recursive structure or operation; $S_i$, $1 \leq i \leq n$, is a set that is equivalent to a type in computing for elements $e$, in which all $e$ share the property $p_i(e)$. 

