Chapter 15
The Formal Design Model of Doubly-Linked-Circular Lists (DLC-Lists)

Yingxu Wang
University of Calgary, Canada

Cyprian F. Ngolah
Sentinel Trending & Diagnostics Ltd., Calgary, Canada

Xinming Tan
Wuhan University of Technology, China

Phillip C.-Y. Sheu
University of California, Irvine, USA

ABSTRACT

Abstract Data Types (ADTs) are a set of highly generic and rigorously modeled data structures in type theory. Lists as a finite sequence of elements are one of the most fundamental and widely used ADTs in system modeling, which provide a standard encapsulation and access interface for manipulating large-volume information and persistent data. This paper develops a comprehensive design pattern of formal lists using a doubly-linked-circular (DLC) list architecture. A rigorous denotational mathematics, Real-Time Process Algebra (RTPA), is adopted, which allows both architectural and behavioral models of lists to be rigorously designed and implemented in a top-down approach. The architectural models of DLC-Lists are created using RTPA architectural modeling methodologies known as the Unified Data Models (UDMs). The behavioral models of DLC-Lists 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 file management system (FMS), and the ADT library for an RTPA-based automatic code generation tool.

1. INTRODUCTION

Data object modeling is a process to creatively extract and abstractly represent a real-world problem by data models. A list is a finite sequence of elements where the order information of its elements is preserved beyond that of sets (Stubbs & Webre, 1985; McDermid, 1991; Bolllella et al., 2002). Many important data objects can be modeled and implemented by lists such as a set of data with keys, a sentence for natural language parsing, a sequential file, a tree, and a graph (Wiener and Pinson, 2000).
A list can be formally modeled by an Abstract Data Type (ADT) (Guttag, 1977; Broy et al., 1984; Cardelli & Wegner, 1985; Stubbs & Webre, 1985), which is an abstract logical model of a complex and/or user defined data structure with a set of predefined operations. Using types to model real-world entities can be traced back to the mathematical thought of Bertrand Russell in 1900s (Russell, 1903) and Georg Cantor in 1932 (Lipschutz & Lipson, 1997). 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: (1) An extension of type constructions by integrating both data structures and functional behaviors; (2) A hybrid data object modeling technique that encapsulates both user defined data structures and allowable operations on them; (3) 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.

In order to formally model the list 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 models and the unified process models (Wang, 2007).

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

\[
UDM \triangleq \prod_{i=1}^{n} (S_i \forall e \in S_i, p_i(e) >)
\]  

where \( S_i, 1 \leq i \leq n \), is a set and also a type of elements \( e \) that share the property \( p_i \).

**Definition 2**: The **Unified Process Model** (UPM) of a program \( \varphi \) is a composition of a finite set of \( m \) processes according to the time-, event-, and interrupt-based process dispatching rules, \( e \_k \rightarrow P_k \), i.e.:

\[
UPM \triangleq \varphi
\]

\[
= \prod_{k=1}^{m} \left( \prod_{i=1}^{n-1} R(s(k) r_j(k) s_j(k)) \right), j = i + 1
\]  

where \( s_i \) and \( s_j \) are one of the 17 RTPA meta-processes, \( r_j \) is one of the 17 RTPA algebraic process operations, and \( e_k \) is a general, timing, or interrupt event.

This paper develops a comprehensive design pattern of formal lists using a doubly-linked-circular (DLC) model. In the remainder of this paper, the conceptual models of lists are described as the initial requirements for describing the list ADT. The architectural model of lists is created as a set of UDMs based on doubly linked and circular nodes dynamically created in the memory. The static behaviors of lists are modeled and refined by a set of UPMs in three categories namely the management operations, traversal operations, and node I/O operations. The dynamic behaviors of lists are formally specified and refined by process scheduling and system dispatching models with UPMs.