Chapter 15
On the Big-R Notation for Describing Iterative and Recursive Behaviors

Yingxu Wang
University of Calgary, Canada

ABSTRACT

Iterative and recursive control structures are the most fundamental mechanisms of computing that make programming more effective and expressive. However, these constructs are perhaps the most diverse and confusable instructions in programming languages at both syntactic and semantic levels. This article introduces the big-R notation that provides a unifying mathematical treatment of iterations and recursions in computing. Mathematical models of iterations and recursions are developed using logical inductions. Based on the mathematical model of the big-R notation, fundamental properties of iterative and recursive behaviors of software are comparatively analyzed. The big-R notation has been adopted and implemented in Real-Time Process Algebra (RTPA) and its supporting tools. Case studies demonstrate that a convenient notation may dramatically reduce the difficulty and complexity in expressing a frequently used and highly recurring concept and notion in computing and software engineering.

INTRODUCTION

A repetitive and efficient treatment of recurrent behaviors and architectures is one of the most premier needs in computing. Iterative and recursive constructs and behaviors are most fundamental to computing because they enable programming to be more effective and expressive. However, unlike the high commonality in branch structures among programming languages, the syntaxes of loops are far more than unified. There is even a lack of common semantics of all forms of loops in modern programming languages (Louden, 1993; Wang, 2006a; Wilson and Clark, 1988).
On the Big-R Notation for Describing Iterative and Recursive Behaviors

When analyzing the syntactic and semantic problems inherited in iterations in programming languages, B. L. Meek concluded that: “There are some who argue that this demonstrates that the procedural approach to programming languages must be inadequate and fatally flawed, and that coping with something so fundamental as looping must therefore entail looking at computation in a different way rather than trying to devise better procedural syntax. There are others who would argue that the possible applications of looping so it cannot simply be removed or obviated. As ever it is probably this last argument that will hold sway until (or unless) someone proves them wrong, whether with a brilliant stroke of procedural syntactic genius, or an effective and comprehensive new approach to the whole area” (Meek, 1991).

This article introduces the big-R notation that provides a unifying mathematical treatment of iterations and recursions in computing. It summarizes the basic control structures of computing, and introduces the big-R notation on the basis of mathematical inductions. The unified mathematical models of iterations and recursions are derived using the big-R notation. Basic properties of iterative and recursive behaviors and architectures in computing are comparatively analyzed. The big-R notation has been adopted and implemented in Real-Time Process Algebra (RTPA) and its supporting tools (Wang, 2002, 2003; Tan, Wang, & Ngolah, 2004). Application examples of the big-R notation in the context of RTPA will be provided throughout this article.

THE BIG-R NOTATION

Although modern high-level programming languages provide a variety of iterative constructs, the mechanisms of iteration may be expressed by the use of conditional or unconditional jumps with a body of linear code. The proliferation of various loop constructs in programming indicates a fundamental need for expressing the notion of repetitive, cyclic, recursive behaviors, and architectures in computing.

In the development of RTPA (Wang, 2002, 2003, 2006a, 2007a, 2007b), it is recognized that all iterative and recursive operations in programming can be unified on the basis of a big-R notation (Wang, 2006b). This section introduces the big-R notation and its mathematical foundation. It can be seen that a convenient notation may dramatically reduce the difficulty and complexity in expressing a frequently used and highly recurring concept and notion in programming.

The Basic Control Structures of Computing

Before the big-R notation is introduced, a survey of essential basic control structures in computing is summarized and reviewed below.

**Definition 1.** Basic control structures (BCS’s) are a set of essential flow control mechanisms that are used for modeling logical architectures of software.

The most commonly identified BCS’s in computing are known as the sequential, branch, case (switch), iterations (three types), procedure call, recursion, parallel, and interrupt structures (Backhouse, 1968; Dijkstra, 1976; Wirth, 1976; Backus, 1978; de Bakker, 1980; Jones, 1980; Cries, 1981; Hehner, 1984; Hoare, 1985, Hoare et al., 1987; Wilson & Clark, 1988; Louden, 1993; Wang, 2002; Horstmann & Budd,
Related Content

From Engineer to Architecture? Designing for a Social Constructivist Environment
[www.igi-global.com/chapter/engineer-architecture-designing-social-constructivist/6628?camid=4v1a](www.igi-global.com/chapter/engineer-architecture-designing-social-constructivist/6628?camid=4v1a)

A Fixpoint Semantics for Rule-Base Anomalies
[www.igi-global.com/chapter/fixpoint-semantics-rule-base-anomalies/27313?camid=4v1a](www.igi-global.com/chapter/fixpoint-semantics-rule-base-anomalies/27313?camid=4v1a)

Materialized View Selection Using Set Based Particle Swarm Optimization

Fuzzy Neural Network Control for Robot Manipulator Directly Driven by Switched Reluctance Motor
[www.igi-global.com/article/fuzzy-neural-network-control-robot/60744?camid=4v1a](www.igi-global.com/article/fuzzy-neural-network-control-robot/60744?camid=4v1a)