Chapter 10
Rosetta Composition Semantics

Megan Peck
University of Kansas, USA

Perry Alexander
University of Kansas, USA

ABSTRACT

The Rosetta specification language aims to enable system designers to abstractly design complex heterogeneous systems. To this end, Rosetta allows for compositional design to facilitate modularity, separation of concerns, and specification reuse. The behavior of Rosetta components and facets can be viewed as systems, which are well suited for coalgebraic denotation. The previous semantics of Rosetta lacked detail in the denotational work, and had no firm semantic basis for the composition operators. This thesis refreshes previous work on the coalgebraic denotation of Rosetta. It then goes on to define the denotation of the composition operators. Several Rosetta examples using all types of composition serve as a demonstration of the power of composition as well as the clean modular abstraction it affords the designer.

INTRODUCTION

System-level design is characterized by the need to assemble information from multiple, heterogeneous domains when making design decisions. Languages intending to support system-level design must therefore support defining and composing multiple models representing heterogeneous information. To achieve this, such languages must eschew single semantics approaches and instead provide support for modeling and composition across multiple semantics. Following this approach one can reason about smaller pieces within an appropriate domain vocabulary and semantics, rather than trying to apply automated reasoning to a monolithic specification, then compose results to make a system-wide correctness assessment (Frisby et al., 2011).

DOI: 10.4018/978-1-4666-4494-6.ch010

The Rosetta specification language (Alexander, 2006; Alexander et al., 2000) provides a language and semantics in support of system-level design. Rosetta focuses on heterogeneous model composition in its domain-based specification system and model composition system. Like traditional hardware description languages, it allows the designer to compose systems from components, or decompose a system into its parts. Unlike traditional hardware description languages, Rosetta supports concurrent engineering by allowing multiple models to describe a system simultaneously. Rosetta's built-in composition operators give the designer the ability to compose components and systems from domain-specific pieces describing different system aspects or components.

The basic specification building block in a Rosetta specification is the *component* that collects a set of declarations and states assumptions, definitions, and implications about those declarations within a single domain. In Rosetta, components are first-class structures and can be manipulated as data. Here we describe three operation classes defined for component composition: structural composition constructs a component that includes the operand facets as components; conjunctive composition defines a component that satisfies all given operand components; and disjunctive composition defines a component that satisfies one or more of the given operand facets.

Formalizing the semantics of a specification language gives us assurances as to the validity of the specifications we write with the language. This work refreshes previous work on the coalgebraic denotation of components and facets. While the ideas of the Rosetta composition operators are not new, they have yet to be formally denoted. Here we fill that semantic void by defining the denotation of the composition operators.

BACKGROUND

By their nature, complex systems must be viewed from many different perspectives. A building provides comfort (HVAC, plumbing, heating, lighting and aesthetics), shelter (structure and safety), and consumes resources (power, water, maintenance) in addition to supporting its business function. Similarly, an embedded system has weight, must be manufactured, must operate in real-time, generates electromagnetic noise, and consumes power in addition to performing its basic function. These systems requirements are as important as a systems basic function. For example, no one buys a cell phone because it makes phone calls – battery life, weight, interoperability, and smart functions play a huge role in such purchases. Given the importance of these system-level issues, design languages should support their inclusion as first-class citizens in the design process.

Rosetta accommodates these multiple specification views by providing a framework for heterogeneous specification, called *domains* (Streb et al., 2006; Streb and Alexander, 2006). Each domain defines a modeling vocabulary and semantics for representing information related to some specification viewpoint. The basic building blocks of Rosetta are *components* and *facets* use domains to represent system models. A *component* defines assumptions, definitions, and implications over a set of declarations specifying not only behavior, but assumptions on that behavior and secondary behaviors that follow from the specification. A *facet* is a component with no assumptions or implications – a basic, ideal model. Every facet or component models a specific system aspect by extending a domain with specific definitions for the system being modeled. For example, an encryption device might be specified in the discrete_time domain while a filter might be written