Chapter 6
Mappings of MOF Metamodels and Algebraic Languages

INTRODUCTION

In this chapter we examine the relation between NEREUS and formal specification using CASL (Common Algebraic Specification Language) as a common algebraic language (Bidoit & Mosses, 2004).

CASL is an expressive and simple language based on a critical selection of known constructs such as subsorts, partial functions, first-order logic, and structured and architectural specifications. A basic specification declares sorts, subsorts, operations and predicates, and gives axioms and constraints. Specifications are structured by means of specification building operators for renaming, extension and combining. Architectural specifications impose structure on implementations, whereas structured specifications only structure the text of specifications.

CASL allows loose, free and generated specifications. The models of a loose specification include all those where the declared functions have the specified properties, without any restrictions on the set of values corresponding to the various sorts. In models of a generated specification, in contrast, it is required that all values can be expressed by terms formed from the specified constructors, i.e. unreachable values are prohibited. In models of free specifications, it is required that values of terms are distinct except when their equality follows from the specified axioms: the possibility of unintended coincidence between their axioms is prohibited.

DOI: 10.4018/978-1-61520-649-0.ch006
CASL is at the center of a family of specification languages. It has restrictions to various sublanguages, and extensions to higher-order, state-based, concurrent, and other languages. CASL is supported by tools and facilitates interoperability of prototyping and verification tools.

Algebraic languages do not follow similar structuring mechanisms to UML or NEREUS. The graph structure of a class diagram involves cycles such as those created by bidirectional associations. However, the algebraic specifications are structured hierarchically and cyclic import structures between two specifications are avoided. In the following, we describe how to translate basic specification in NEREUS into CASL, and then we analyze how to translate associations (Favre, 2009), (Favre, 2006) (Favre, 2005).

**TRANSLATING BASIC SPECIFICATIONS**

In NEREUS the elements of `<parameterList>` are pairs C1:C2 where C1 is the formal generic parameter constrained by an existing class C2 or C1: ANY. In CASL, the first syntax is translated into `[C2]` and the second in `[sort C1]`. Next, we show two expressions in NEREUS and the CASL:

**NEREUS**

CLASS CartesProd [ E:ANY; E1: ANY]

**CASL**

spec CARTESPROD [sort E] [sort E1]

CLASS HASH [T:ANY; V:HASHABLE]

**CASL**

spec HASH [sort T] [HASHABLE]

NEREUS and CASL have the similar syntax for declaring types. The sorts in the IS-SUBTYPE paragraph are linked to subsorts in CASL.

The signatures of the NEREUS operations are translated into operations or predicates in CASL. Data type declarations may be used to abbreviate declarations of types and constructors.

Any NEREUS function that includes partial functions must specify the domain of each of them. This is the role of the PRE clause that indicates what conditions the function’s arguments must satisfy to belong to the function’s domain. To indicate that a CASL function may be partial the notation uses -›?; the normal arrow will be reserved for total functions. The translation includes an axiom for restricting the domain. For example, a partial function remove (see Bidirectional-3 specification in Appendix-B)

remove: Bidirectional-3 (b) x Class1 (c1) x Class2(c2) -› Bidirectional-3

pre: isRelated (b,c1,c2)

is translated into

remove: Bidirectional-3 (b) x Class1 x Class2 -›? Bidirectional-3

... 
forall b:Bidirectional-3, c1:Class1; c2: Class2
def remove (b,c1,c2) ⊳ isRelated (b,c1,c2)