Introduction

- **Purpose**
  - Specification of external interfaces
    - Operations (arguments, results)

- **Example**
  - Abstract data types
    - You define behavior of all the operations, and not the internal data representation

- **Usage**: prototyping
  - Executable specifications
Algebraic method

- Using
  - Algebraic structures
  - Abstract data types

- ADT = carrier sets + operations + axioms
Basic theory
Algebra

- Algebra $A = \langle D, F \rangle$
  - Carrier set $D$
  - Functions $F$

- Function $f_A \in F$
  - $f_A : A \times \ldots \times A \rightarrow A$
  - $f_A : \rightarrow A$
Sorts

- Sort = data type
  - Examples: Nat, Int, Bool, Strings, ...

- Many-sorted algebras

- Sub-sorting relation
  - Nat < Int
Notation
- S ... sorts
- F ... functions (operations)
- D ... carrier sets (data)
- A ... algebra

Types of functions
- T = S* × S
- s₁ × ... × sₙ → s

Algebra A = <[Dₛ]{s ∈ S}, [Fₜ]{t ∈ T}>
Example
Signature

- Signature \((S, \Sigma)\)
  - \(\Sigma = [\Sigma_t]_{t \in T}\)

- \(\Sigma\)-algebra
  - Carrier set \(D_s\) for every sort \(s \in S\)
  - Operation \(f_A\) for each symbol \(f \in F\)
Properties of operations

• Basic approach
  - Equations between function expressions

• Set E of all equations (sentences, axioms)

• Executable specifications (models)
More complex signatures and equations

- Overloaded functions
  - Different subsorts
  - Number of arguments

- Predicates and relations
  - Signature: the set P of predicate symbols
Initial model

- Exactly the right number of elements in carrier sets
  - No redundancy ("garbage")
  - No ambiguity ("confusion")

- Multiple isomorphic models
Algebraic specification

• Assumptions
  ▫ Programs are modeled by many-sorted algebras
  ▫ Correctness of the input/output behavior has precedence over all other properties

• \( Q = (S, \Sigma, E) \)

• Two parts
  ▫ Declarations (signature)
  ▫ Equations (semantics)
Example

- List of integers
  - Operations: add, remove, get, size, contains
    - insert and remove to/from any position

- Use of recursion

- Constructing bigger instances (values)
  - Lists with multiple elements

- Exceptions (errors)
Semantics of algebraic specifications

- $Q = (S, \Sigma, E)$
  - well-formed specification

- $\text{Sem}[Q]$
  - the class of all initial algebras (models)
Languages

- CASL: Common Algebraic Specification Language
  - http://www.cofi.info

- Other: Larch (family), OBJ3, ASL
• Ian Sommerville: Software Engineering
  - consider just recent book editions (9th or 10th)