

# CS3773 Software Engineering

## Lecture 13 Algebraic Specification

## Specifying Data Types

- We want to specify interfaces of components to facilitate system decomposition
- We need to specify interesting data types sometimes
  - Sets
  - Arrays
  - Records
  - New data types with operations
- We want to avoid making design decisions, such as how a data type is implemented (algorithm and data structure)

2

UTSA CS3773

## Prose Specification of Data Types

- Strategy #1: write a prose specification
  - A stack of integers supports three operations: push, pop, and size. Pushing adds an element. Popping returns the most recently added element. Size returns the number of elements.*
- Problems: it is ambiguous
  - What happens when you pop an empty stack?
  - Does "return" imply removes as well or just peeks?
  - How to add elements?
  - What are the precise signatures of the operations?

3

UTSA CS3773

## Structured Prose of Data Types

- Strategy #2: write a careful and structured prose
- That is what most specifications look like
- Problems
  - Suffers from the same problem as prose

4

UTSA CS3773

## Operational Specification of Data Types

- Strategy #3: give an operational specification, i.e., write some code as the specification
  - Formal
  - Lots of supporting tools, such as compilers
- Problems
  - Over constraining
  - Committing to representation and algorithm strategy
    - Java ints are 4 bytes
    - ...
  - Too much inessential detail

5

UTSA CS3773

## Algebraic Specification

- ADT = Abstract Data Type
  - Sort: set of values (an abstract view of data)
  - Description: informal specification which is easy to understand
  - Literals: constants of the data type
  - Signature: operators' names, domains, and ranges
- Algebraic specification = ADT + axioms
- Algebraic specification describes
  - Name of the sort
  - Signatures of operators
  - Properties of the operators as relationships between operators

6

UTSA CS3773

## An Example: A Stack of Integers

```
NEWTYPE IntStack
LITERALS EmptyStack;
OPERATORS
  push: Integer. IntStack -> IntStack;
  pop: IntStack -> IntStack;
  top: IntStack -> Integer;
  isEmpty: IntStack -> Boolean;
AXIOMS
  FOR ALL s in IntStack (
    FOR ALL i in Integer (
      pop (EmptyStack) == EmptyStack;
      pop (push (i,s)) == s;

      top (EmptyStack) == error!;
      top (push (i,s)) == i;

      isEmpty (EmptyStack) == true;
      isEmpty (push (i,s)) == false;
    )
  );
ENDNEWTYPE
```

7

UTSA CS3773

## An Example: A Stack of Integers

- All stacks start out empty, and then have various operations performed on them
- What is returned after a sequence of function calls on the IntStack ADT
  - new; push 5; push 8; top  
applying axiom #4  
top (push (8, push (5, ES))) = ?
  - new; push 3; push 7; push 9; pop; pop  
applying axiom #2  
pop (pop (push (9, push (7, push (3, ES)))))) = ?

8

UTSA CS3773

## Advantages of Algebraic Specification

- Algebraic specification has no implementation bias  
no description of how to represent a stack
- We specify the interrelationships between operations  
rather than each individually
- Multiple implementations can satisfy the specification

9

UTSA CS3773

## Values of ADT

- Values of an ADT are expressed abstractly as sequences of operations on the literals, and these sequence are called terms or traces
  - Multiple terms may represent the same value
  - Terms represent equivalent values can be grouped into equivalence classes
  - The axioms describe which terms are equivalent
  - There may be a minimal, unique term for a value

10

UTSA CS3773

## Classes of Operations

- **Generators**
  - Minimal set of operations needed to construct any value of the data type
  - Building canonical representation of the defined data type
- **Manipulators**
  - Returning values of the defined data type
  - Not generators
- **Inspectors**
  - Evaluating attributes of the sort
  - Not returning values of the defined data type

11

UTSA CS3773

## Canonical Terms

- The minimal representation of a value of data type is called a canonical term or a canonical trace
- A canonical trace contains only a sequence of generators
- In the IntStack specification
  - Eliminating all the pop operations results in a unique, minimal representation of a value of type IntStack
  - Any stack's history can be rewritten to remove all pushes and pops of elements not in the stack

12

UTSA CS3773

## Disadvantages of Algebraic Specification

- Algebraic specifications are hard to read and write
- Algebraic specifications are hard to change
- It is hard to tell if your algebraic specification is
  - Correct
  - Complete
  - Consistent

13

UTSA CS3773

## Process for Building Algebraic Specification

- Determine what operations and literals you need
- Categorize operations and literals into three groups
  - Generators
  - Manipulators
  - Inspectors
- Define result of applying non-generator operations after each generator operation

14

UTSA CS3773

## Reading Assignments

- Sommerville's Book
  - Chapter 10, "Formal Specification"
    - 10.2 Sub-system interface specification

15

UTSA CS3773