By Sunday:

Download and checkout SML for New Jersey

Read through chapter 3 in book

**ADT (abstract data type)** -- One ADT definition will be correct for many implementations

Define the behavior once then:

it guides implementation

**Abstractions** -- leave out a lot of details when modeling something that don’t matter to the question at hand.

\*\*Look at *Model of UNC*\*\*

***How can Data be abstract?***

We think of it as data values together with behavior… operations that manipulate the data values. So for every operation, we want to define a function that will transform its arguments into return values.

***To make data abstract:***

**Leave out:** details related to implementation in any particular programming language

**Leave In**: changes made to the state of the data

First, develop the functional signature (type of user interface)

Next, provide an axiomatic specification of the behavior of each operation (method)

**Stack**: (LIFO - last one in first one out) A stack is an order of data. You can access the top only.

*new() --* creates new stack

*push(73)* -- ads data (73) to top of stack

pop() -- throw away top of stack and make it smaller by 1

size() -- size of stack

**Specifying ADT**

*Guttag’s Method:*

**Example: *STACK of Int***

**new**: → creates a new **STACK**

**push**: STACK x Int → passes back a **STACK**

**pop**: STACK → passes back **STACK**

**top**: STACK → returns the **INT** on top of the *STACK*

**size**: STACK → returns a natural number **NAT**

**Axioms of Behavior**

Idea is to write an equation (axium) giving two expivlent forms of the data structure

Axioms

Ex: size( new() ) = 0

Ex: size( push( new(), 6) ) = 1

Ex: top( push( push( new(), 3), -8) ) = -8

Ex: pop( new() ) = new()

Ex: pop( push( S, i)) = S

Ex: top( new() ) = err

The ability to create true statements is infinite. The above is all true, but they don’t capture everything

**Guttag’s Method:**

Look at operations and pick out canonical ones

Make a list of all LHS for axioms by applying each non-canonical op to a canonical op

Create an equivalent RHS for each LHS

A canonical operation is one that is needed if your goal is to generate ALL possible stack values by calling successive operations

A non-canonical op is one that is not needed. In other words it can be replaced by some other operation to get the same answer.

Ex: push( pop( push( new (), 6) ), 3) = 3

push( new, 3) = 3 \*\*Same thing\*\*

*\*A single axiom is the long, one lined expression*

*\*The axiom is an abstraction of the functions (the axiom does not say what happens, because it isn’t necessary, but it still does happen)*

*\*canonical operations GENERATE something*