Lecture 21: Algebraic Specifications

Kenneth M. Anderson

Foundations of Software Engineering

CSCI 5828 - Spring Semester, 1999

Today's Lecture

- Examine Algebraic Specifications
  - – Compare Stack and Queue
  - – Introduce Homework 4

Algebraic Specifications

- Algebras are Akin to Abstract Data Types
- Sets of Values
- Operations
- Many Formalisms
  - – Larch, CCS, Lotos, …
  - – RAISE can be used in an algebraic "style"

Terminology

- Homogeneous Algebra

  Single set and its operations
- Heterogeneous Algebra

  Multiple sets and their operations
- Signature

  Collection of sets in heterogeneous algebra
- Sort

  A set within an algebra

Terminology

- Syntax

  Signature plus operations with domains and ranges

- Semantics

  Equations involving operations; axioms

- Generators

  Operations that create instance of an algebra; inductive rules of inference

Algebraic Specification of Stack

```
algebra StackOfItem
```

Algebraic Specification of Stack

```
algebra StackOfItem
    imports Boolean;
```

Algebraic Specification of Stack

```
algebra StackOfItem
    imports Boolean;
    introduces
        sorts Stack, Item;
```

Algebraic Specification of Stack

```
algebra StackOfItem
    imports Boolean;
    introduces
        sorts Stack, Item;
        operations
            Create: → Stack;
            IsEmpty: Stack → Boolean;
            Push: Stack × Item → Stack;
            Pop: Stack → Stack;
            Top: Stack → Item;
```

Algebraic Specification of Stack

```
algebra StackOfItem
    imports Boolean;
    introduces
        sorts Stack, Item;
        operations
            Create: → Stack;
            IsEmpty: Stack → Boolean;
            Push: Stack × Item → Stack;
            Pop: Stack → Stack;
            Top: Stack → Item;
    constrains Create, IsEmpty, Push, Pop, Top so that
        Stack generated by [Create, Push]
```

Algebraic Specification of Queue

```
algebra QueueOfItem
```

Algebraic Specification of Queue

```
algebra QueueOfItem
    imports Boolean;
```

Algebraic Specification of Queue

```
algebra QueueOfItem
    imports Boolean;
    introduces
        sorts Queue, Item;
```

Algebraic Specification of Queue

```
algebra QueueOfItem
    imports Boolean;
    introduces
        sorts Queue, Item;
        operations
            Create: → Queue;
            IsEmpty: Queue → Boolean;
            Enqueue: Queue × Item → Queue;
            Dequeue: Queue → Queue;
            Front: Queue → Item;
```

Algebraic Specification of Queue

```
algebra QueueOfItem
    imports Boolean;
    introduces
        sorts Queue, Item;
        operations
            Create: → Queue;
            IsEmpty: Queue → Boolean;
            Enqueue: Queue × Item → Queue;
            Dequeue: Queue → Queue;
            Front: Queue → Item;
    constrains Create, IsEmpty, Enqueue, Dequeue, Front so that
        Queue generated by [Create, Enqueue]
```

Algebraic Specification of Pizza

```
algebra Nonsense
    imports Boolean;
    introduces
        sorts Pizza, Car;
        operations
            Cat: → Pizza;
            Horse: Pizza → Boolean;
            Dog: Pizza × Car → Pizza;
            Bird: Pizza → Pizza;
            Mouse: Pizza → Car;
    constrains Cat, Horse, Dog, Bird, Mouse so that
        Pizza generated by [Cat, Horse]
```

Algebraic Specification of Stack

```
algebra StackOfItem
    imports Boolean;
    introduces
        sorts Stack, Item;
        operations
            Create: → Stack;
            IsEmpty: Stack → Boolean;
            Push: Stack × Item → Stack;
            Pop: Stack → Stack;
            Top: Stack → Item;
    constrains Create, IsEmpty, Push, Pop, Top so that
        Stack generated by [Create, Push]
```

Algebraic Specification of Stack

```
    for all [s: Stack; i: Item]







end StackOfItem;
```

Algebraic Specification of Stack

```
    for all [s: Stack; i: Item]
        IsEmpty(Create) = true;






end StackOfItem;
```

Algebraic Specification of Stack

```
    for all [s: Stack; i: Item]
        IsEmpty(Create) = true;
        IsEmpty(Push(s,i)) = false;





end StackOfItem;
```

Algebraic Specification of Stack

```
    for all [s: Stack; i: Item]
        IsEmpty(Create) = true;
        IsEmpty(Push(s,i)) = false;
        Pop(Create) = error;



end StackOfItem;
```

Algebraic Specification of Stack

```
    for all [s: Stack; i: Item]
        IsEmpty(Create) = true;
        IsEmpty(Push(s,i)) = false;
        Pop(Create) = error;
        Top(Create) = error;


end StackOfItem;
```

Algebraic Specification of Stack

```
    for all [s: Stack; i: Item]
        IsEmpty(Create) = true;
        IsEmpty(Push(s,i)) = false;
        Pop(Create) = error;
        Top(Create) = error;
        Pop(Push(s,i)) = s;

end StackOfItem;
```

Algebraic Specification of Stack

```
    for all [s: Stack; i: Item]
        IsEmpty(Create) = true;
        IsEmpty(Push(s,i)) = false;
        Pop(Create) = error;
        Top(Create) = error;
        Pop(Push(s,i)) = s;
        Top(Push(s,i)) = i;
end StackOfItem;
```

Algebraic Specification of Queue

```
algebra QueueOfItem
    imports Boolean;
    introduces
        sorts Queue, Item;
        operations
            Create: → Queue;
            IsEmpty: Queue → Boolean;
            Enqueue: Queue × Item → Queue;
            Dequeue: Queue → Queue;
            Front: Queue → Item;
    constrains Create, IsEmpty, Enqueue, Dequeue, Front so that
        Queue generated by [Create, Enqueue]
```

Algebraic Specification of Queue

```
    for all [q: Queue; i: Item]









end QueueOfItem;
```

Algebraic Specification of Queue

```
    for all [q: Queue; i: Item]
        IsEmpty(Create) = true;









end QueueOfItem;
```

Algebraic Specification of Queue

```
    for all [q: Queue; i: Item]
        IsEmpty(Create) = true;
        IsEmpty(Enqueue(q,i)) = false;









end QueueOfItem;
```

Algebraic Specification of Queue

```
for all [q: Queue; i: Item]
    IsEmpty(Create) = true;
    IsEmpty(Enqueue(q,i)) = false;
    Dequeue(Create) = error;

end QueueOfItem;
```

Algebraic Specification of Queue

```
for all [q: Queue; i: Item]
    IsEmpty(Create) = true;
    IsEmpty(Enqueue(q,i)) = false;
    Dequeue(Create) = error;
    Front(Create) = error;

end QueueOfItem;
```

Algebraic Specification of Queue

```
for all [q: Queue; i: Item]
    IsEmpty(Create) = true;
    IsEmpty(Enqueue(q,i)) = false;
    Dequeue(Create) = error;
    Front(Create) = error;
    Dequeue(Enqueue(q,i))

end QueueOfItem;
```

Algebraic Specification of Queue

```
for all [q: Queue; i: Item]
    IsEmpty(Create) = true;
    IsEmpty(Enqueue(q,i)) = false;
    Dequeue(Create) = error;
    Front(Create) = error;
    Dequeue(Enqueue(q,i)) = if (IsEmpty(q))

end QueueOfItem;
```

Algebraic Specification of Queue

```
   for all [q: Queue; i: Item]
      IsEmpty(Create) = true;
      IsEmpty(Enqueue(q,i)) = false;
      Dequeue(Create) = error;
      Front(Create) = error;
      Dequeue(Enqueue(q,i)) = if (IsEmpty(q))
                                             then Create




end QueueOfItem;
```

Algebraic Specification of Queue

```
   for all [q: Queue; i: Item]
      IsEmpty(Create) = true;
      IsEmpty(Enqueue(q,i)) = false;
      Dequeue(Create) = error;
      Front(Create) = error;
      Dequeue(Enqueue(q,i)) = if (IsEmpty(q))
                                             then Create
                                             else Enqueue(Dequeue(q),i);



end QueueOfItem;
```

Algebraic Specification of Queue

```
   for all [q: Queue; i: Item]
      IsEmpty(Create) = true;
      IsEmpty(Enqueue(q,i)) = false;
      Dequeue(Create) = error;
      Front(Create) = error;
      Dequeue(Enqueue(q,i)) = if (IsEmpty(q))
                                             then Create
                                             else Enqueue(Dequeue(q),i);
      Front(Enqueue(q,i))


end QueueOfItem;
```

Algebraic Specification of Queue

```
   for all [q: Queue; i: Item]
      IsEmpty(Create) = true;
      IsEmpty(Enqueue(q,i)) = false;
      Dequeue(Create) = error;
      Front(Create) = error;
      Dequeue(Enqueue(q,i)) = if (IsEmpty(q))
                                             then Create
                                             else Enqueue(Dequeue(q),i);
      Front(Enqueue(q,i)) = if (IsEmpty(q))


end QueueOfItem;
```

Algebraic Specification of Queue

```
     for all [q: Queue; i: Item]
         IsEmpty(Create) = true;
         IsEmpty(Enqueue(q,i)) = false;
         Dequeue(Create) = error;
         Front(Create) = error;
         Dequeue(Enqueue(q,i)) = if (IsEmpty(q))
                                     then Create
                                     else Enqueue(Dequeue(q),i);
         Front(Enqueue(q,i)) = if (IsEmpty(q))
                                  then i

 end QueueOfItem;
```

Algebraic Specification of Queue

```
     for all [q: Queue; i: Item]
         IsEmpty(Create) = true;
         IsEmpty(Enqueue(q,i)) = false;
         Dequeue(Create) = error;
         Front(Create) = error;
         Dequeue(Enqueue(q,i)) = if (IsEmpty(q))
                                     then Create
                                     else Enqueue(Dequeue(q),i);
         Front(Enqueue(q,i)) = if (IsEmpty(q))
                                  then i
                                  else Front(q);
 end QueueOfItem;
```

Homework 4

- Give the semantics for an algebraic specification of a set of items
  - I give you the syntax
- Sets contain only one instance of a particular value
  - e.g. Adding {2} to {1, 2} gives {1, 2}
  - Adding {3} to {1, 2} gives {1, 2, 3}