

Algebraic Abstract Data Types: Introduction and Syntax

Didier Buchs

Université de Genève

23 septembre 2021

Algebraic Abstract Data Types

- Informal introduction
- AADT Signature
- Terms with variables
- Equations and axioms
- Examples
- Graceful presentations
- Examples

Formal and Mathematical basis

- Algebraic view
 - heterogeneous algebra (Birkhoff) = sets + operations
 - Logical view of their properties (Horn clauses)
- Computer science
 - Type = set of data + operations
 - Some code for describing the behavior of these types
- Support of an Abstraction point of view
 - Information hiding (realization hiding)
 - Functional approach (data hiding)

Informal example : Manipulation of strings

- Mandatory operations :
 - An empty string (new)
 - Concatenation of two strings (append)
 - Concatenation of one character to the string (add to)
 - Computation of the length (size)
 - Test of emptiness (isEmpty?)
 - Equality of two strings (=)
 - Selection of the first element (first)
- Necessary types for defining the string abstract data type :
 - character : the character AADT
 - natural : the type of the natural numbers
 - boolean : the type of the boolean values

Signature

Definition of set of values and operations = signatures

- signatures
 - sorts names (or types)
 - operations names with profile (arity) nameofoperation :
domain -> co-domain

Adt StringSpec;

Interface

sorts string, character, natural, boolean;

Operations

new: () -> string;

append _ _: string, string -> string;

add _ to _: character, string -> string;

size _ : string -> natural;

isEmpty? _ : string -> boolean;

_ = _: string, string -> boolean;

first _ : string -> character;

Remarks on the syntax : generalized prefix,infix and postfix notations

Prefix :

append _ _ : string, string -> string;

constructible terms

append x y

append(x y)

(append x y)

Remarks on the syntax(2)

Infix :

$_ = _ : \text{string, string} \rightarrow \text{boolean};$

constructible terms

$x = y$

$(x = y)$

Remarks on the syntax(3)

Mixfix :

add _ to _: character, string -> string;

constructible terms

add append(x y) to c

add c to append(x y)

add first(x) to y

shows / term w/ hyp̄

Remarks on the signature

Terminology :

- string is the sort of interest
- character, natural et boolean are auxiliary sorts

Observation operations :

```
_ = _ : string, string -> boolean;  
size _ : string -> natural;  
isEmpty? _ : string -> boolean;  
first _ : string -> character;
```

Definition (Observer)

An observer is an operation with the profile :
interest sort and ev. auxiliary sorts → auxiliary sort

Remarks on the signature(2)

Modifier operations :

```
new: () -> string;  
add _ to _: character, string-> string;  
append _ _: string, string -> string;
```

Definition (Modifier)

A modifier is an operation with the profile :
interest sort and ev. auxiliary sorts → interest sort

A subclass of modifier is the operations generating all values of the domain.

Definition (Generator)

A generator is an operation with the profile :
interest sort and ev. auxiliary sorts → interest sort

Definition of basic set concepts

We recall here some usual definitions.

- **S** be universe of all sort names (type names).
- Universe are used to provide disjoint domains for sets
- two different universe are disjoint

Definition (Disjoint Union)

a disjoint union is a union where elements of the union are always considered different i.e. $\forall A, B$ sets,

$$A' = A \times \{0\}, A' \cong A$$

$$B' = B \times \{1\}, B' \cong B$$

$$A \amalg B \Leftrightarrow A' \cup B'$$

Example :

Definition of S-sorted set

We give here some basic definitions for typing objects.

Definition (S-Sorted Set)

Let $S \subseteq \mathbf{S}$ be a finite set. A S -sorted set A is a disjoint union of a family of sets indexed by S ($A = \coprod_{s \in S} A_s$), noted as $A = (A_s)_{s \in S}$.

Remark : In our theory this is a disjoint partition, for non-disjoint partition there is theory of ordered sorts.

Example :

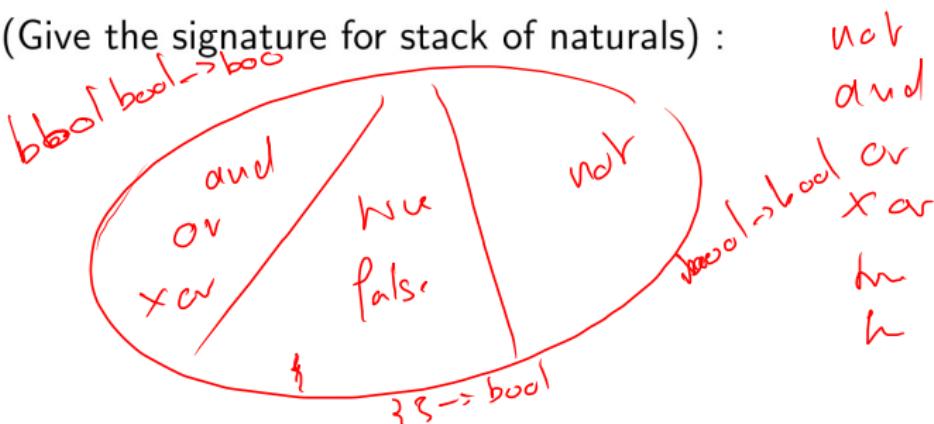
Definition of signature

Based on S-sets we have :

Definition (Signature)

A *signature* is a couple $\Sigma = \langle S, F \rangle$, where $S \subseteq \mathbf{S}$ is a finite set of sorts and $F = (F_{w,s})_{w \in S^*, s \in S}$ is a $(S^* \times S)$ -sorted set of function names of \mathbf{F} . Each $f \in F_{\epsilon,s}$ is called a *constant*.

Example (Give the signature for stack of naturals) :



Definition of terms

Definition (Terms of a Signature)

Let $\Sigma = \langle S, F \rangle$ be a signature and X be a S -sorted set of variables. The set of terms of Σ over X is a S -sorted set $T_{\Sigma, X}$, where each set $(T_{\Sigma, X})_s$ is inductively defined as follows :

- each variable $x \in X_s$ is a term of sort s , i.e., $x \in (T_{\Sigma, X})_s$
- each constant $f \in F_{\epsilon, s}$ is a term of sort s , i.e., $f \in (T_{\Sigma, X})_s$
- for all operations that are not a constant $f \in F_{w, s}$, with $w = s_1 \dots s_n$, and for all n -tuple of terms $(t_1 \dots t_n)$ such that all $t_i \in (T_{\Sigma, X})_{s_i}$ ($1 \leq i \leq n$), $f(t_1 \dots t_n) \in (T_{\Sigma, X})_s$

What means this term ?

add c to x = append(x y)
append (IsEmpty(new), add x to c)

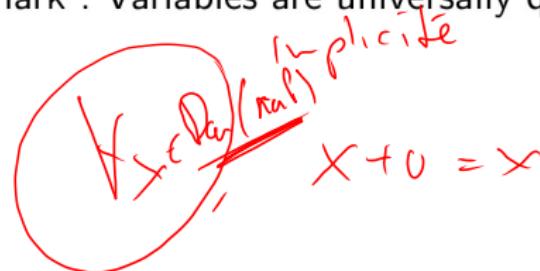
Definition of axioms

Definition (Axioms on variables)

Let $\Sigma = \langle S, F \rangle$ be a signature and X be a S -sorted set of variables. The *axioms on variables* X are equational terms $t = t'$ such that $t, t' \in (T_{\Sigma, X})_s$.

Example : $x+0 = x$

Remark : Variables are universally quantified



A handwritten note in red ink. It starts with a large red circle containing the text "forall x in Var (real)". An arrow points from this circle to the term "x + 0 = x" located to the right. Above the arrow, the word "implicite" is written.

$$\forall x \in \text{Var}(\text{real}) \text{ implicate } x + 0 = x$$

String Axioms

Axioms

```
isEmpty?(new) = true;
isEmpty?(add c to x) = false;
#( new) = 0;
#(add c to x) = # (x) + 1;
append(new, x) = x;
append(add c to x, y) = add c to (append( x,y));
(new = new) = true;
(add c to x = new) = false;
(new = add c to x) = false;
(add c to x = add d to y) = (c = d) and (x = y);
+ axioms of first
```

Where

```
x,y:string; c,d:character;
End StringSpec;
```

String Axioms(2)

Be carefull!! : The symbol = is either a signature operator and a meta-operator of the basic logic of the specification language.

Signature of auxiliary sorts :

```
true: -> boolean;  
false: -> boolean;  
not _ : boolean -> boolean;  
_ and_ ,_ or_ : boolean, boolean -> boolean ;  
0: -> natural; 1: -> natural;  
succ: natural -> natural;  
_ + _ ,_ -_ ,_ *_ ,_ /_ : natural, natural -> natural ;  
_ =_ : natural, natural -> boolean;  
a: () -> character; b: () -> character;  
....  
_ =_ : character, character -> boolean;
```

Boolean Axioms

```

Adt Booleans;
Interface
Sorts boolean;
Operations
true , false : -> boolean;
not _ : boolean -> boolean;
_ and _ ,_ or _ , _ xor _ ,_ = _ : boolean boolean -> boolean
Body
Axioms
not(true) = false; not(false) = true;
(true and b) = b; (false and b) = false;
(true or b) = true; (false or b) = b;
(false xor b) = b; (true xor b) = not(b);
(true = true) = true; (true = false) = false;
(false = true) = false; (false = false) = true;
Where b : boolean;

```

$\text{X and False} = \text{False}$

$\text{X and True} = \text{X}$

Exercise

Write the axioms of a sort Stack with the signature ;

Adt Stack;

Interface

Use Naturals, Booleans;

Sorts stack;

Operations

empty : \rightarrow stack;

push _ _ : natural stack \rightarrow stack;

pop _ : stack \rightarrow stack;

top _ : stack \rightarrow natural;

_ = _ : stack stack \rightarrow boolean;

Exercise

Axioms of a sort Stack :

Conditional Axioms

Positive conditional axioms (Horn clause with equality) :

Definition (Axioms on variables)

Let $\Sigma = \langle S, F \rangle$ be a signature and X be a S -sorted set of variables. The *conditional axioms on variables X* are

$t_1 = t'_1 \wedge \dots \wedge t_n = t'_n \Rightarrow t = t'$ such that

$t, t' \in (T_{\Sigma, X})_s, t_1, t'_1 \in (T_{\Sigma, X})_{s_1}, \dots, t_n, t'_n \in (T_{\Sigma, X})_{s_n}, ..$

```
isEmpty(x)= false =>
first(add c to x) = first x;
isEmpty(x)= true =>
first(add c to x) = c;
```

Is it necessary ?

Graceful presentations

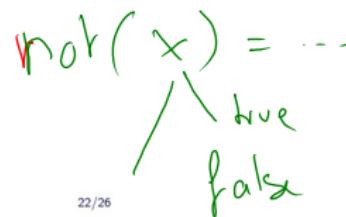
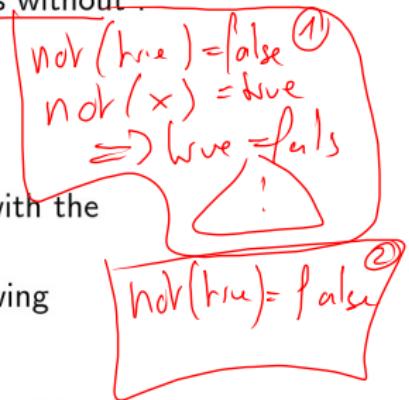
Graceful presentations It is a method for writing axioms without :

- the possibility of writing contradictory axioms
- forgetting cases.

Principle for each operation of the signature :

- Write on the left of the equation a term starting with the name of this operation.
- Iterate on all parameter of the operation the following principle from left to right :
 - Use a variable for this parameter
 - If it is not possible to write a valid axiom for this variable decompose using the generators
 - If a generator is not sufficient for the decomposition in sub case use conditions

General Property : sufficient completeness and hierarchical consistency are guaranteed



Example of axiomatisation

$$x + y = ?$$

natwell

zero : $\rightarrow \text{nat}$

succ : $\text{nat} \rightarrow \text{nat}$

decomposition of the second parameter with both constructors!

$$0 + X = X + 0$$

① $x + 0 = x;$

$$x + \text{succ}(y) = \text{succ}(x+y);$$

Exercise : Application to

$$x > y = ?$$

$$x > \text{succ}(y) = \text{false} \quad 0 + \text{succ}(0) = \text{succ}(0) + 0 \\ \text{succ}(0 + 0) = \underline{\text{succ}(0)}$$

$$\boxed{x > 0}$$

② $0 > 0 = \text{false}$

③ $(\text{succ}(y) > 0) = \text{true}$

④ $(x > \text{succ}(y)) = \text{true}$

⑤ $(x > \text{succ}(y)) = \text{true}$

$$7 > 3$$

$$S^7(0) > S^3(0)$$



$$S^6(0) > S^2(0)$$



$$S^5(0) > S(0)$$



$$S^4(0) > 0$$



Example of axiomatisation : Sets of naturals.

$$\{a, b\} = \{b, a\} \cup \{a\}$$

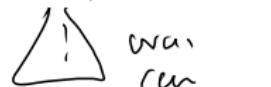
	Sort	Set
gen	empty	\rightarrow set
op	add	: set, nat \rightarrow set
	in	: set, nat \rightarrow bool
	union	: set, set \rightarrow set
	size	\rightarrow nat

works

$$\begin{aligned} \text{add}(\text{add}(\text{empty}, 0), 1) &= \\ \text{add}(\text{add}(\text{empty}, 1), 0) & \\ \text{add}(\text{add}(\text{empty}, 1), 1) & \\ = \text{add}(\text{empty}, 1) & \end{aligned}$$

~~axioms~~ ① $\text{add}(\text{add}(s, n), m) = \text{add}(\text{add}(s, m), n)$

② $\text{add}(\text{add}(s, n), n) = \text{add}(s, n)$



warn
rem
axiom

$\text{add}(\text{empty}, m) = \text{add}(\text{empty}, m)$ ③ order

24/26

$\text{in}(\text{empty}, n) = \text{false}$

~~$\text{in}(\text{add}(s, n), n) = \text{true}$~~

$\text{eq}(n, m) = \text{true} \Rightarrow \text{in}(\text{add}(s, n), m) = \text{true}$

$\text{eq}(n, m) = \text{false} \Rightarrow \text{in}(\text{add}(s, n), m) = \text{in}(s, m)$

$\text{size}(\text{empty}) = 0$

$\text{in}(s, n) = \text{false} \Rightarrow \text{size}(\text{add}(s, n)) = \text{succ}(\text{size}(s))$

$\text{in}(s, n) = \text{true} \Rightarrow \text{size}(\text{add}(s, n)) = \text{size}(s)$

$\text{union}(\text{empty}, s) = s$

$\text{union}(\text{add}(s, t), n) = \text{add}(\text{union}(s, t), n)$

$\text{in}(\text{empty}, s) = \text{empty}$

$\text{in}(t, n) = \text{true}$
 $\Rightarrow \text{in}(\text{add}(s, n), t) = \text{odd}(\text{in}(s, t), n)$

$\text{in}(t, n) = \text{false}$

$\Rightarrow \text{in}(\text{add}(s, n), t) = \text{in}(s, t)$

Example of axiomatisation : Tables of naturals

$\text{empty} : \longrightarrow \text{table}$
 $-[-] := - ; \text{table}, \text{index}, \text{value} \rightarrow \text{table}$

$$\text{eq}(i, j) = \text{false} \Rightarrow ((t[i] := v) [j] := w) =$$

$$((t[j] := w) [i] := v)$$

$$\text{eq}(i, i) = \text{true}$$

$$((\delta[i] := v) [j] := w) =$$

$$+ [i] := w$$

Summary

- Sorts and functions
- Axioms
- Graceful Presentation
- Subtleties when algebras are not free