Program Verification: Lecture 3

José Meseguer

Computer Science Department
University of Illinois at Urbana-Champaign

Algebras

An (unsorted, many-sorted, or order-sorted) signature Σ is just syntax: provides the symbols for a language; but what is that language talking about? what is its semantics?

It is obviously talking about algebras, which are the mathematical models in which we interpret the syntax of Σ, giving it concrete meaning.

Unsorted algebras are the simplest example: children become familiar with them from the early awakenings of reason. They consist of a set of data elements, and various chosen constants among those elements, and operations on such data.

Algebras (II)

For example, for Σ the unsorted signature of the module `NAT-MIXFIX` we can define many different algebras, such as the following:

1. $\mathbb{N}$, the algebra of natural numbers in whatever notation we wish (Peano, binary, base 10, etc.) with `0` interpreted as the zero element, `s` interpreted as successor, and `_+_` and `_*_` interpreted as natural number addition and multiplication.

2. $\mathbb{N}_k$, the algebra of residue classes modulo k, for k a nonzero natural number. This is a finite algebra whose set of elements can be represented as the set $\{0, \ldots, k-1\}$. We interpret `0` as 0, and for the other

operations we perform them in $\mathbb{N}$ and then take the residue modulo k. For example, in $\mathbb{N}_7$ we have $6+6=5$.

3. $\mathbf{Z}$, the algebra of the integers, with 0 interpreted as the zero element, s interpreted as successor, and _+_ and _*_ interpreted as integer addition and multiplication.

4. $\mathbf{Q}$, the algebra of the rational numbers, with 0 interpreted as the zero element, s interpreted as adding 1, and _+_ and _*_ interpreted as rational addition and multiplication.

5. $\mathbf{R}$, the algebra of the real numbers, with 0 interpreted as the zero element, s interpreted as adding 1, and _+_ and _*_ interpreted as real number addition and multiplication.

6. $\mathbf{C}$, the algebra of the complex numbers, with 0

interpreted as the zero element, `s` interpreted as adding 1, and `_+_` and `_*_` interpreted as complex number addition and multiplication.

Similarly, for Σ the unsorted signature:

```
sort Boolean .
ops true false : -> Boolean .
op not : Boolean -> Boolean .
ops and or : Boolean Boolean -> Boolean .
```

we can define many algebras, including the following:

1. $\mathbf{B}$ the standard Boolean algebra, with just two elements, say $\{0, 1\}$, with `true` interpreted as 1 and `false` as 0 and with the standard interpretation of `not`, `and`, and `or` as Boolean operations.

2. (powersets) for X any set, we can view its powerset $\mathcal{P}(X)$ as a Σ-algebra for this signature, with `true` interpreted as X, `false` as $\emptyset$, `not` interpreted as complement (that is, $not(Y) = X - Y$), and with `and`, and `or` interpreted, respectively, as set intersection $\cap$ and set union $\cup$.

3. (Boolean predicates on a set) for X any set, we can endow the function set $[X \to \mathbf{B}]$ with a Σ-algebra structure for this signature, with `true` interpreted as the constant function 1, `false` as the constant function 0, $\texttt{not}(f)$ interpreted as the function $not(f)(x) = not_{\mathbf{B}}(f(x))$, where $not_{\mathbf{B}}$ is the intepretation in the standard Boolean algebra $\mathbf{B}$ above, and with $f\,\texttt{and}\,g$ and $f\,\texttt{or}\,g$ interpreted, respectively, as the functions: $(f\,\texttt{and}\,g)(x) = f(x) and_{\mathbf{B}} g(x)$, $(f\,\texttt{or}\,g)(x) = f(x) or_{\mathbf{B}} g(x)$.

Note that, up to a slight change of representation (viewing a predicate as a subset or, alternatively, as a Boolean-valued function) for any set X the algebras $\mathcal{P}(X)$ and $[X{\rightarrow}\mathbf{B}]$ are essentially the same algebra, that is, they are isomorphic, a notion that will be further explained later in the course.

4. (Fuzzy truth values) The set of data elements is the real interval $[0, 1]$. We interpret `true` as 1, `false` as 0, $not(x) = 1 - x$, `and` as minimum, and `or` as maximum.

Definition of Unsorted Algebras

For Σ an unsorted signature $\Sigma = (\{s\}, F)$, an unsorted Σ-algebra is a pair $\mathcal{A} = (A, \_\mathcal{A})$, where A is a set, specifying the data elements in the algebra, and $\_\mathcal{A} = \{f_\mathcal{A}\}_{(f,w,s)\in F}$ is an F-indexed set called the interpretation function that maps:

- each constant $a :\longrightarrow s$ in F to an element $a_\mathcal{A} \in A$
- each n-ary function symbol $f : s \overset{n}{\ldots} s \longrightarrow s$ in F to a function $f_\mathcal{A} : A^n \longrightarrow A$.

Definition of Many-Sorted Algebras

Similarly, for $\Sigma = (S, F)$ a many-sorted signature, a many-sorted Σ-algebra is a pair $\mathcal{A} = (A, \_\mathcal{A})$, with $A = \{A_s\}_{s \in S}$ an S-indexed family of sets, specifying the data elements for each sort $s \in S$, and $\_\mathcal{A} = \{f_\mathcal{A}\}_{(f,w,s) \in F}$ an F-indexed set mapping:

- each constant $a : \longrightarrow s$ in F to an element $a_\mathcal{A} \in A_s$
- each function symbol $f : s_1 \ldots s_n \longrightarrow s$ in F to a function $f_\mathcal{A} : A_{s_1} \times \ldots \times A_{s_n} \longrightarrow A_s$.

Notation: if $w = s_1 \ldots s_n$, we write $A^w = A_{s_1} \times \ldots \times A_{s_n}$.

Examples of Many-Sorted Algebras

For Σ the signature of the module `NAT-LIST` we can define several algebras:

1. (Strings of naturals). We interpret the sort `Natural` as the set $\mathbb{N}$ of natural numbers, and the sort `List` as the set of strings $\mathbb{N}^*$. The interpretation function for the constants and operations is then as follows: (i) all operations in the submodule `NAT-MIXFIX` are intepreted as the algebra $\mathbb{N}$ of natural numbers; (ii) `nil` is interpreted as the empty string; (iii) `_._` is interpreted as the function that concatenates a natural number on the left of a string; and (iv) `length` is interpreted as the function measuring the length of a string.

2. (Sets of naturals). We interpret the sort `Natural` as the set $\mathbb{N}$ of natural numbers, and the sort `List` as the set

$\mathcal{P}_{fin}(\mathbb{N})$ of finite subsets of $\mathbb{N}$. The interpretation function for the constants and operations is then as follows: (i) all operations in the submodule NAT-MIXFIX are intepreted as the algebra $\mathbb{N}$ of natural numbers; (ii) nil is interpreted as the empty set $\emptyset$; (iii) _._ is interpreted as the function inserting a natural number on a set of naturals; and (iv) length is interpreted as the cardinality function.

For another series of examples, consider the many-sorted signature Σ in Picture 3.1. The following are then examples of Σ-algebras:

1. (n-dimensional rational, real, and complex vector spaces). The sort Scalar is interpreted, respectively, by $\mathbf{Q}$, $\mathbf{R}$, $\mathbf{C}$; and the sort Vector by, respectively, $\mathbf{Q}^n$, $\mathbf{R}^n$, $\mathbf{C}^n$. The operations of sort Scalar are interpreted on,

repectively, $\mathbf{Q}$, $\mathbf{R}$, and $\mathbf{C}$, exactly as in the signature of `NAT-MIXFIX` as already explained. The only new constant now is 1, which is intepreted precisely as the number 1 in all cases. Vector addition is intepreted in all three cases in the usual way:

$$(x_1, \ldots, x_n) + (y_1, \ldots, y_n) = (x_1 + y_1, \ldots, x_n + y_n)$$

The constant $\vec{0}$ is interpreted as the zero vector $(0, \overset{n}{\ldots}, 0)$. The operation symbol $\_.\_$ is intepreted by the rule: $\lambda.(x_1, \ldots, x_n) = (\lambda * x_1, \ldots, \lambda * x_n)$.

2. (n-dimensional integer modules). Exactly as above, but using $\mathbf{Z}$ as scalars, and $\mathbf{Z}^n$ as vectors.

3. (n-dimensional natural semi-modules). Exactly as above, but using $\mathbb{N}$ as scalars, and $\mathbb{N}^n$ as vectors.

4. (n-dimensional natural-modulo-k semi-modules). Exactly as above, but using $\mathbb{N}_k$ as scalars, and $\mathbb{N}_k^n$ as vectors.

5. (n-dimensional natural-modulo-k semi-modules with natural scalars). Use $\mathbb{N}$ as scalars, and $\mathbb{N}_k^n$ as vectors.

6. We can also use a smaller algebra for scalars. For example, we can choose $\mathbf{C}^n$ as vectors, and either $\mathbf{R}$, $\mathbf{Q}$, $\mathbf{Z}$, or $\mathbb{N}$ as scalars. Likewise, we can use $\mathbf{R}^n$ as vectors, and either $\mathbf{Q}$, $\mathbf{Z}$, or $\mathbb{N}$ as scalars. Similarly, we can use $\mathbf{Q}^n$ as vectors, and either $\mathbf{Z}$, or $\mathbb{N}$ as scalars. Finally, we can use $\mathbf{Z}^n$ as vectors, and $\mathbb{N}$ as scalars.

7. (Fuzzy (sub-)sets). A subset $Y \subseteq X$ can be characterized a Boolean-valued predicate $p_Y : X \longrightarrow \mathbf{B}$, with $p_Y(x) = if\ x \in Y\ then\ 1\ else\ 0\ fi$. By definition, a

fuzzy subset of X is a function $Z : X \longrightarrow [0, 1]$. Z is a "subset with uncertainty" about which elements belong to it, so that its boundaries become "fuzzy" (see Picture 3.2). The set of fuzzy subsets of X is $[X{\rightarrow}[0, 1]]$. We can define a Σ-algebra by choosing $[X{\rightarrow}[0, 1]]$ as the interpretation of the `Vector` sort, wih vector addition interpreted as union of fuzzy sets, where, by definition, $(Z \cup U)(x) = max(Z(x), U(x))$, and $\vec{0}$ interpreted as the empty fuzzy set (the constantly zero function). For the scalars we choose the interval $[0.1]$ with 0 interpreted as 0, 1 as 1, $\_ + \_$ as maximum, and $\_ * \_$ as minimum. Scalar multiplication is defined by the rule: $(\lambda.Z)(x) = min(\lambda, Z(x))$. Alternatively, we could instead define a different scalar multiplication by: $(\lambda.Z)(x) = \lambda * Z(x)$; this would give us a different Σ-algebra structure on fuzzy sets.

Definition of Order-Sorted Algebras

Given an order-sorted signature $\Sigma = ((S, <), F)$ an order-sorted Σ-algebra is defined as a many-sorted (S, F)-algebra $\mathcal{A} = (A, \_{}_{\mathcal{A}})$ such that:

- In $A = \{A_s\}_{s \in S}$, if $s < s'$ then $A_s \subseteq A_{s'}$
- if f is subsort overloaded, so that we have, $f : w \longrightarrow s$, and $f : w' \longrightarrow s'$, with w and w' strings of equal length, and with $w \equiv_{\leq} w'$ and $s \equiv_{\leq} s'$, then:
  - if $w = w' = nil$, then f is a constant and we have $f_{\mathcal{A}}^{nil,s} = f_{\mathcal{A}}^{nil,s'}$ (subsort overloaded constants coincide)
  - otherwise, if $(a_1, \ldots, a_n) \in A^w \cap A^{w'}$, then $f_{\mathcal{A}}^{w,s}(a_1, \ldots, a_n) = f_{\mathcal{A}}^{w',s'}(a_1, \ldots, a_n)$ (subsort overloaded operations agree)

Examples of Order-Sorted Algebras

For Σ the signature of `NAT-LIST-II` we can define, among others, two different order-sorted algebra structures:

1. Interpret the sort `NzNatural` as $\mathbb{N}^{\bullet}$, `Natural` as $\mathbb{N}$, s, p, and $\_ + \_$ in the usual way, `NeList` as $\mathbb{N}^{+}$, `List` as $\mathbb{N}^{*}$, nil as the empty string, $\_.\_$ as left concatenation with a natural, and $first$, $rest$ and $length$ in the usual way.

2. We can instead interpret both `NzNatural` and `Natural` as $\mathbf{Z}$, s, p, and $\_ + \_$ as those functions extended to all integers, `NeList` as $\mathbf{Z}^{+}$, `List` as $\mathbf{Z}^{*}$, nil is as the empty string, $\_.\_$ as left concatenation with an integer, and $first$, $rest$ and $length$ in the usual way.

Term Algebras

An obvious example of an order-sorted Σ-algebra is the term algebra $\mathcal{T}_\Sigma = (T_\Sigma, \iota_{\mathcal{T}_\Sigma})$, where the family $T_\Sigma = \{T_{\Sigma,s}\}_{s \in S}$ and its operations are mutually defined by:

- for each $a : nil \longrightarrow s$ in Σ, $a_{\mathcal{T}_\Sigma} = a \in T_{\Sigma,s}$
- for each $f : w \longrightarrow s$ in Σ, with $w = s_1 \ldots s_n$, $n > 0$, the function $f_{\mathcal{T}_\Sigma} : T_{\Sigma,s_1} \times \ldots \times T_{\Sigma,s_n} \longrightarrow T_{\Sigma,s}$ maps the tuple $(t_1, \ldots, t_n) \in T_\Sigma^w$ to the expression (called a term) $f(t_1, \ldots, t_n) \in T_{\Sigma,s}$
- if $s < s'$, then $T_{\Sigma,s} \subseteq T_{\Sigma,s'}$

Examples of Terms for the NATURAL Specification

$T_{\texttt{NATURAL},\texttt{NzNatural}} =$
$\{\texttt{s 0}, \texttt{s s 0}, \texttt{s s s 0}, \texttt{s p s 0}, \texttt{s(0 + s 0)}, \ldots\}$

$T_{\texttt{NATURAL},\texttt{Natural}} =$
$T_{\texttt{NATURAL},\texttt{NzNatural}} \cup \{\texttt{0}, \texttt{p s 0}, (0 + 0), \ldots\}.$

Although the mathematical definition of terms uses prefix notation, Maude allows general mixfix notation. This is just a (very useful) parsing and pretty-printing facility. If one insists (by giving the command `set print mixfix off .`) Maude can print even mixfix terms in prefix notation. For example, `s_(_+_(0,s_(0)))` instead of `s(0 + s 0)`.

Sensible Signatures

A signature Σ can be intrinsically ambiguous, so that a term may denote two completely different things. Consider for example the following signature:

```
sorts A B C D .
op a : -> A .
op f : A -> B .
op f : A -> C .
op g : B -> D .
op g : C -> D .
```

then the term `g(f(a))` is an ambiguous term of sort `D` denoting two completely different things.

A very mild condition ruling this out, yet allowing ad-hoc overloading, is the notion of a sensible signature, namely one such that whenever we have $f : w \longrightarrow s$ and $f : w' \longrightarrow s'$, with w, w' of equal length, then $w \equiv_{\leq} w' \Rightarrow s \equiv_{\leq} s'$.

Sensible Signatures (II)

Lemma. If Σ is a sensible order-sorted signature, then for any term t in T_Σ we have,

$$t \in T_{\Sigma,s} \;\wedge\; t \in T_{\Sigma,s'} \quad\Rightarrow\quad s \equiv_{\leq} s'$$

Proof: By induction on the depth of t.

We define the depth of a term as follows: constants have depth 0, and terms of the form $f(t_1, \ldots, t_n)$ have depth $1 + max(depth(t_1), \ldots, depth(t_n))$.

For depth 0, $t = a$ is a constant, and $a \in T_{\Sigma,s}$ iff there is $a : nil \to s''$ in Σ with $s'' \leq s$. Similarly, if $a \in T_{\Sigma,s'}$ there is $a : nil \to s'''$ in Σ with $s''' \leq s'$. By Σ sensible we have $s'' \equiv_{\leq} s'''$, and therefore, $s \equiv_{\leq} s'$.

Sensible Signatures (III)

Assuming the result true for depth $\leq n$, let $t = f(t_1, \ldots, t_n)$ have depth $n + 1$. If we have $t \in T_{\Sigma,s} \ \wedge \ t \in T_{\Sigma,s'}$, this forces the existence of $f : w'' \longrightarrow s''$ and $f : w''' \longrightarrow s'''$, with $s'' \leq s$ and $s''' \leq s'$ and such that $(t_1, \ldots, t_n) \in T_{\Sigma}^{w''} \cap T_{\Sigma}^{w'''}$.

By the induction hypothesis this forces $w'' \equiv_{\leq} w'''$. And by Σ sensible this forces $s'' \equiv_{\leq} s'''$, and therefore, $s \equiv_{\leq} s'$. q.e.d.

Variables

Note that in our definition of Σ-terms we only allowed constants and terms built up from them by other operation symbols, so-called ground terms. Therefore, terms with variables, such as those appearing in the equations

```
vars N M : Natural .
eq N + 0 = N .
eq N + s M = s(N + M) .
```

do not seem to fall within our definition. What can we say about such terms? First, note that `N` and `M` are variables in the mathematical sense, not at all in the sense of variables in an imperative language. Second, we can reduce the notion of terms with variables to that of terms without variables (ground terms) in an extended signature.

A Sample Extended Signature

We can extend the signature of our above example by adding the variables as additional constants to get the new signature,

```
sort Natural .
op 0 : -> Natural .
op N : -> Natural .
op M : -> Natural .
op s_ : Natural -> Natural .
op _+_ : Natural Natural -> Natural .
```

in which a term such as `s(N + M)` is now a well-defined term of sort `Natural`.

The Extended Signature $\Sigma(X)$

The general way of extending a signature $\Sigma = ((S, <), F)$ with variables is as follows. We assume a family $X = \{X_s\}_{s \in S}$ of sets of variables for the different sorts $s \in S$ in the signature Σ. Such that:

- variables of different sorts are different, i.e., $X_s \cap X_{s'} = \emptyset$ if $s \neq s'$
- the variables in X are different from the constants in Σ, i.e., $(\cup_{s \in S} X_s) \cap \{(a, nil, s) \in F \mid s \in S\} = \emptyset$.

Then we define the extended signature $\Sigma(X) = ((S, <), F(X))$, where $F(X) = F \cup \{(x, nil, s) \mid x \in X_s \ \wedge \ s \in S\}$.

The Term Algebra $\mathcal{T}_{\Sigma(X)}$

Therefore, Σ-terms with variables in X are the elements of the term algebra $\mathcal{T}_{\Sigma(X)}$ associated to the extended signature $\Sigma(X)$.

Note that if Σ is a sensible signature, then it is trivial to check that $\Sigma(X)$ is also, by construction, a sensible signature. Therefore, all the results holding for ground terms in sensible signatures do hold likewise for terms with variables.

Preregular Signatures

An order-sorted signature $\Sigma = ((S, <), F)$ is called preregular iff for each Σ-term t (possibly with variables), the set of sorts

$$\{s \in S \mid t : s\}$$

has a least element in the poset $(S, <)$ called the least sort of t and denoted $ls(t)$.

Maude automatically checks the preregularity of the signature Σ of any module entered by the user and issues a warning if Σ is not preregular.

Indexed Families of Functions

Given two S-indexed families of sets $A = \{A_s\}$, and $B = \{B_s\}$, what is the natural generalization of the concept of function from sets to S-indexed sets?

It is the notion of an S-indexed family of functions (or S-indexed function) $f = \{f_s : A_s \longrightarrow B_s\}_{s \in S}$, where for each $s \in S$ $f_s : A_s \longrightarrow B_s$ is a function in the ordinary sense. We then write $f : A \longrightarrow B$.

Mathematically, S-indexed functions $f = \{f_s : A_s \longrightarrow B_s\}_{s \in S}$ can be described as ordinary functions $f : S \longrightarrow \mathcal{U}$ into the set $\mathcal{U} = \bigcup_{s \in S} [A_s \to B_s]$, such that for each $s \in S$ we have $f(s) \in [A_s \to B_s]$.

Substitutions

Given an order-sorted signature Σ with set of sorts S and given S-indexed families of variables $X = \{X_s\}_{s \in S}$, and $Y = \{Y_s\}_{s \in S}$, a substitution is an S-indexed family of functions

$$\theta : X \longrightarrow T_{\Sigma(Y)}$$

For example, for Σ an unsorted signature of arithmetic expressions, $X = \{x, y, z\}$, and $Y = \{x, y, z, x', y', z'\}$, a particular θ can be the assignment:

- $x \mapsto (x + y') * z$
- $y \mapsto (x' - y')$
- $z \mapsto z' * z'$

Substitutions Extend to Terms

If Σ is a sensible signature, a substitution

$$\theta : X \longrightarrow T_{\Sigma(Y)}$$

extends in a unique way to an S-indexed function

$$\_\theta : T_{\Sigma(X)} \longrightarrow T_{\Sigma(Y)}$$

defined recursively by:

- $x\theta = \theta(x)$
- $f(t_1, \ldots, t_n)\theta = f(t_1\theta, \ldots, t_n\theta)$

For example, for the above θ we have,

$$x + (y * z)\theta = ((x + y') * z) + ((x' - y') * (z' * z')).$$

Equations

What is the general notion of equation in the general setting of Σ-algebras that we are considering?

Given a signature Σ, we can define a Σ-equation as an expression of the form

$$t = t'$$

where, for some variables X, t and t' are terms $t \in T_{\Sigma(X)_s}$ and $t' \in T_{\Sigma(X)_{s'}}$ such that the sorts s, s' are related in the subsort ordering, i.e., $s \equiv_{\leq} s'$.

Given the above equation, we can universally quantify all its variables and get the sentence

$$(\forall X)\; t = t'$$

Some Common Mistakes

- not ending declarations for sorts, operators, etc. with a space followed by a period, e.g.,

  ```
  sort Natural
  op 0 : -> Natural.
  op s : Natural -> Natural
  ```

- not putting enough parentheses to disambiguate expressions, e.g., `p s s 0 + 0`

- not leaving spaces between a mixfix operator and its arguments, e.g., 0+0

Getting to Use Maude

You should begin writing functional modules of your own with syntax as exemplified in the above examples. You should write such modules in files using Emacs.

Download Maude from the Maude web page http://maude.cs.uiuc.edu. Read Setion 1.7 of "All About Maude" for suggestions on how beginners can become acquainted with Maude as soon as possible.

To enter a module into Maude can use cut and paste, or the "in *filename*" command inside Maude, and can change or list directories using Unix commands.

Readings and Exercises

Before the next lecture try to:

- Follow the reading suggestions for beginners in 1.7 of "All About Maude," and try to get as deep as possible this way into Chapter 4.

- Continue playing with Maude. Define other functions on commonly used data types. For example, define binary trees that have natural numbers in their leaves, and define three functions: (i) tree **reverse**, (ii) **max** and **min** (give the biggest, resp. smallest, number stored in the tree), and (iii) **insert**, which inserts a number in the tree, so that numbers to its left in the tree will be smaller.

Readings and Exercises (II)

Ex.3.1. Denote by $C_1, \ldots C_n, \ldots$ the different connected components of the poset of sorts in a signature Σ. Also, given a sort s, denote by $C(s)$ its connected component. Then, given a Σ-algebra $\mathcal{A} = (A, \_\mathcal{A})$ and a connected component C define $A_C = \cup_{s \in C} A_s$.

Prove that, given an operator $f : s_1 \ldots s_n \longrightarrow s$ in Σ, all the other subsort-overloaded operators related to it "glue together" into a single partial function $f_{\mathcal{A}} : A_{C(s_1)} \times \ldots \times A_{C(s_n)} \longrightarrow A_{C(s)}$.

Therefore, the qualifications $f_{\mathcal{A}}^{w,s}$ are in some sense unnecessary.

Readings and Exercises (III)

Ex.3.2. Let Σ be the signature:

```
sort Natural .
op 0 : -> Natural .
op s : Natural -> Natural .
```

And let $A = \{a, b, c\}$. How many different Σ-algebra structures can be defined on the set A? That is, how many different Σ-algebras of the form $\mathcal{A} = (A, \_{\mathcal{A}})$ are there? (Explain, and also state the total number of such algebras). Can you justify why the number comes out that way? For example, can your supposed justification predict (without having to explicitly construct them) exactly how many such algebras will there be on A if we add to the above Σ a binary function, say,

```
op _+_ : Natural Natural -> Natural .
```