Program Verification: Lecture 9

José Meseguer

Computer Science Department
University of Illinois at Urbana-Champaign

Unsorted Homomorphisms

Given unsorted Σ-algebras $\mathcal{A} = (A, \_{\mathcal{A}})$ and $\mathcal{B} = (B, \_{\mathcal{B}})$, a Σ-homomorphism h from $\mathcal{A}$ to $\mathcal{B}$, written $h : \mathcal{A} \longrightarrow \mathcal{B}$, is a function $h : A \longrightarrow B$ such that (with s the only sort) we have:

- for each constant $a : nil \longrightarrow s$ in Σ, $h(a_{\mathcal{A}}) = a_{\mathcal{B}}$ (preservation of constants)

- for each $f : s \overset{n}{\ldots} s \longrightarrow s$ in Σ and each $(a_1, \ldots, a_n) \in A^n$, we have $h(f_{\mathcal{A}}(a_1, \ldots, a_n)) = f_{\mathcal{B}}(h(a_1), \ldots, h(a_n))$ (preservation of operations)

Examples of Unsorted Homomorphisms

The term algebra $\mathcal{T}_{\Sigma_{\texttt{NAT-MIXFIX}}}$, the natural numbers $\mathbb{N}$, and the natural numbers modulo n, $\mathbb{N}_n$ (for any $n \geq 1$) are all $\Sigma_{\texttt{NAT-PREFIX}}$-algebras (Lectures 2–3). Show that (for any n) we have $\Sigma_{\texttt{NAT-PREFIX}}$-homomorphisms:

$$\mathcal{T}_{\Sigma_{\texttt{NAT-PREFIX}}} \xrightarrow{\_\mathbb{N}} \mathbb{N} \xrightarrow{rem_n} \mathbb{N}_n$$

where $\_\mathbb{N}$ evaluates a term to its value in $\mathbb{N}$, and rem_n sends each number to its remainder after dividing by n. For example, we should have:

- $(s(0) + s(0))_{\mathbb{N}} = 2$, and
- $rem_7(23) = 2$.

Show that $\_\mathbb{N}; rem_n$ is also a homomorphism, and that we have the identity $\_\mathbb{N}; rem_n = \_\mathbb{N}_n$.

Examples of Unsorted Homomorphisms (II)

Recall (Lecture 2, pg. 30) the powerset algebra $\mathcal{P}(X)$ over the Boolean signature Σ_{BOOL}. Let X and Y be any sets, and let $f : X \longrightarrow Y$ be any function. Prove in detail that the function

$$f^{-1} : \mathcal{P}(Y) \longrightarrow \mathcal{P}(X)$$

defined for any $A \subseteq Y$ by: $f^{-1}(A) = \{x \in X \mid f(x) \in A\}$, is a Σ_{BOOL}-homomorphism. Prove also that if we also have a function $g : Y \longrightarrow Z$, then we have the identity $(f; g)^{-1} = g^{-1}; f^{-1}$, and therefore that $g^{-1}; f^{-1} : \mathcal{P}(Z) \longrightarrow \mathcal{P}(X)$ is also a Σ_{BOOL}-homomorphism.

Many-Sorted Homomorphisms

Given (many-sorted) Σ-algebras $\mathcal{A} = (A, \_{}_{\mathcal{A}})$ and $\mathcal{B} = (B, \_{}_{\mathcal{B}})$, a Σ-homomorphism h from $\mathcal{A}$ to $\mathcal{B}$, written $h : \mathcal{A} \longrightarrow \mathcal{B}$, is an S-indexed family of functions $h = \{h_s : A_s \longrightarrow B_s\}_{s \in S}$ such that:

- for each constant $a : nil \longrightarrow s$, $h_s(a_{\mathcal{A}}^{nil,s}) = a_{\mathcal{B}}^{nil,s}$ (preservation of constants)
- for each $f : w \longrightarrow s$ with $w = s_1 \ldots s_n$, $n \geq 1$, and each $(a_1, \ldots, a_n) \in A^w$, we have $h_s(f_{\mathcal{A}}^{w,s}(a_1, \ldots, a_n)) = f_{\mathcal{B}}^{w,s}(h_{s_1}(a_1), \ldots, h_{s_n}(a_n))$ (preservation of operations)

Examples of Many-Sorted Homomorphisms

Recall the module `NAT-LIST` in Lecture 2, and the two algebras on such a signature, let us call them $\mathcal{A}$ and $\mathcal{B}$, defined on page 34–35 of Lecture 2, namely $\mathcal{A}$ = lists of natural numbers and $\mathcal{B}$ = (finite) sets of natural numbers. Show that there cannot be any $\Sigma_{\texttt{NAT-LIST}}$-homomorphim $h : \mathcal{A} \longrightarrow \mathcal{B}$.

For Σ the signature in picture 2.4, consider the first family of algebras for it described in point 1, pages 35–36 of Lecture 2, namely n-dimensional vector spaces on the rational, the real, or the complex numbers. Let us be specific and fix the reals. Let $\mathcal{A}$ be the 3-dimensional real vector space, and $\mathcal{B}$ the 2-dimensional real vector space. What is then a Σ-homomorphism $h : \mathcal{A} \longrightarrow \mathcal{B}$? Prove that any such homomorphism h can be completely described by a 2×3 matrix M_h with real coefficients, so that applying to a 3-dimensionsl vector $\vec{v}$ the homomorphims h, that is, computing $h(\vec{v})$ exactly corresponds to computing the matrix multiplication $\vec{v} \circ M_h$. Generalize this to $\mathcal{A}$ and $\mathcal{B}$ real vector spaces of arbitrary finite dimensions n and m. Generalize it further to rational, resp. complex, vector spaces of any pair of finite dimensions n and m.

Now generalize this even further to characterize by means of matrices all Σ-homomorphims between Σ-algebras in cases 2–7 in pages 36–38 of Lecture 2, where in case 7 (fuzzy sets) you sould restrict yourself to the fuzzy subsets of finite sets. Give for each of these cases specific examples of $h : \mathcal{A} \longrightarrow \mathcal{B}$ showing how this works and how h is thus applied to specific elements in the corresponding algebra $\mathcal{A}$.

Order-Sorted Homomorphisms

For $\Sigma = ((S, <), F)$ an order-sorted signature, and $\mathcal{A}$ and $\mathcal{B}$ order-sorted Σ-algebras, a Σ-homomorphism h from $\mathcal{A}$ to $\mathcal{B}$, written $h : \mathcal{A} \longrightarrow \mathcal{B}$, is an S-indexed family of functions $h = \{h_s : A_s \longrightarrow B_s\}_{s \in S}$ such that:

- $h : \mathcal{A} \longrightarrow \mathcal{B}$ is a many-sorted (S, F)-homomorphism; and
- if $[s] = [s']$ and $a \in A_s \cap A_{s'}$, then $h_s(a) = h_{s'}(a)$ (agreement on data in the same connected component)

Examples of Order-Sorted Homomorphisms

Consider the order-sorted signature Σ of the `NAT-LIST-II` exampe in Lecture 2, the two algebras on such a signature, let us call them $\mathcal{A}$ and $\mathcal{B}$, defined on page 40 of Lecture 2, with $\mathcal{A}$ case (1), and $\mathcal{B}$ case (2). Show that there is exactly one order-sorted Σ-homomorphim $h : \mathcal{A} \longrightarrow \mathcal{B}$. Describe such a homomorphism h in complete detail. Show that there cannot be any other Σ-homomorphims $h' : \mathcal{A} \longrightarrow \mathcal{B}$ with $h \neq h'$.

Initiality of the Term Algebra

If a signature is sensible, then different terms denote different things. In the argot of algebraic specifications, this is expressed by saying that the term algebra has no confusion.

Furthermore, the term algebra is in some sense minimal, since it has only the elements it needs to have to be an algebra: the constants, and the terms needed so that the operations can yield a result; that is why this minimality is expressed saying that it has no junk.

Note: In the rest of the course we will always assume that all signatures are sensible.

Initiality of the Term Algebra (II)

This minimality means that there is at most one way to map homomorphically the elements of $\mathcal{T}_\Sigma$ to any algebra. And its "no confusion" lack of ambiguity means that such an homomorphic map can always be defined.

For example, it couldn't be defined for Σ the non-sensible signature we showed in pg. 4 of Lecture 3 and the Σ-algebra $\mathcal{K}$ with: $K_A = \{a\}$, $K_B = \{b\}$, $K_C = \{c\}$, $K_D = \{d, d'\}$, and with $f^{A,B}(a) = b$, $f^{A,C}(a) = c$, $g^{B,D}(b) = d$, and $g^{C,D}(c) = d'$. Indeed, there in no Σ-homomorphism $h : \mathcal{T}_\Sigma \longrightarrow \mathcal{K}$ at all, since $h_D(g(f(a))$ must be either d or d'. If $h_D(g(f(a)) = d$, then h fails to preserve the operation $g : C \longrightarrow D$, and if $h_D(g(f(a)) = d'$, then h fails to preserve the operation $g : B \longrightarrow D$.

Initiality of the Term Algebra (III)

In summary, the claim is that, if Σ is sensible, then for any Σ-algebra $\mathcal{A}$ there is a unique Σ-homomorphism, say, $\_{\mathcal{A}} : \mathcal{T}_\Sigma \longrightarrow \mathcal{A}$. This is called the initiality property of $\mathcal{T}_\Sigma$. The map $\_{\mathcal{A}}$ is the obvious evaluation function, mapping each term t to the result of evaluating it in $\mathcal{A}$. $\_{\mathcal{A}}$ is defined inductively in the obvious way:

- for a constant a we define $(a)_{\mathcal{A}} = a_{\mathcal{A}}$, and
- for a term $f(t_1, \ldots, t_n)$ we define $(f(t_1, \ldots, t_n))_{\mathcal{A}} = f_{\mathcal{A}}((t_1)_{\mathcal{A}}, \ldots, (t_n)_{\mathcal{A}})$.

Let us prove it in detail.

Theorem. If Σ is a sensible order-sorted signature, then $\mathcal{T}_\Sigma$ satisfies the initiality property.

Proof of the Initiality Theorem

Proof: For $\mathcal{A}$ any Σ-algebra Let us first prove the uniqueness of $\_\mathcal{A}$, and then its existence.

Proof of uniqueness. Let us suppose that we have two different homomorphisms $h, h' : \mathcal{T}_\Sigma \longrightarrow \mathcal{A}$. We can prove that $h = h'$ by induction on the depth of the terms.

For terms of depth 0 let a be a constant in $T_{\Sigma,s}$. That means that there is a sort $s' \leq s$ with an operator declaration $a : nil \longrightarrow s'$ and therefore, by h and h' being Σ-homomorphisms we must have $h_s(a) = h'_s(a) = a_{\mathcal{A}}^{nil,s'}$.

Proof of the Initiality Theorem (II)

Assume that the equality $h = h'$ holds for terms of depth less or equal to n, and let $f(t_1, \ldots, t_n) \in T_{\Sigma,s}$ have depth $n+1$. That means that there is an operator declaration $f : s_1 \ldots s_n \longrightarrow s'$ with $s' \leq s$ and $t_i \in T_{\Sigma,s_i}$, $1 \leq i \leq n$. Again, by h and h' being Σ-homomorphisms we must have:

$h_s(f(t_1, \ldots, t_n)) =$
$= f_{\mathcal{A}}^{s_1 \ldots s_n, s'}(h_{s_1}(t_1), \ldots, h_{s_n}(t_n))$ (h homomorphism and $s \leq s'$)
$= f_{\mathcal{A}}^{s_1 \ldots s_n, s'}(h'_{s_1}(t_1), \ldots, h'_{s_n}(t_n))$ (induction hypothesis)
$= h'_s(f(t_1, \ldots, t_n))$ (h' homomorphism and $s \leq s'$).

Proof of the Initiality Theorem (III)

Proof of Existence. We can both define $\_{\mathcal{A}}$ and show that it is a Σ-homomorphism by induction on the depth of terms. For terms of depth 0, let $a \in T_{\Sigma,s}$ be a constant. That means that there is a sort $s' \leq s$ with an operator declaration $a : nil \longrightarrow s'$; we then define $(a)_{\mathcal{A}_s} = a_{\mathcal{A}}^{nil,s'}$.

Note that the constant a could be subsort-overloaded (cannot be ad-hoc overloaded, since this is ruled out by Σ being sensible) but the above assignment is well-defined (does not depend on the particular declaration $a : nil \longrightarrow s'$ chosen), because by our definition of Σ-algebra the interpretations of all subsort overloaded versions of a constant a must coincide in the algebra $\mathcal{A}$. Furthermore, $\_{\mathcal{A}}$ preserves constants, so it is a Σ-homomorphism.

Proof of the Initiality Theorem (IV)

Assume that $\_{\mathcal{A}}$ has already been defined and is a Σ-homomorphism for terms of depth less or equal to n, and let $f(t_1,\ldots,t_n) \in T_{\Sigma,s}$ be a term of depth $n+1$. That means that there is an operator declaration $f : s_1 \ldots s_n \longrightarrow s'$ with $s' \leq s$ and $t_i \in T_{\Sigma,s_i}$, $1 \leq i \leq n$. We define $(f(t_1,\ldots,t_n))_{\mathcal{A}} = f_{\mathcal{A}}^{s_1\ldots s_n,s'}((t_1)_{\mathcal{A}},\ldots,(t_n)_{\mathcal{A}})$.

Note that, by the induction hypothesis, $\_{\mathcal{A}}$ has already been defined for terms of depth less or equal to n and is a Σ-homomorphism on those terms.

Note also that, by the Lemma on sensible signatures, for any other $f : s'_1 \ldots s'_n \longrightarrow s''$ such that $t_i \in T_{\Sigma,s'_i}$, $1 \leq i \leq n$, we must have, $[s_i] = [s'_i]$, $1 \leq i \leq n$, and $[s'] = [s'']$.

Proof of the Initiality Theorem (V)

Since we have $[s_i] = [s'_i]$, $1 \leq i \leq n$, by definition of order-sorted Σ-homomorphism this then forces, $\_\mathcal{A}_{s_i}(t_i) = \_\mathcal{A}_{s'_i}(t_i)$, $1 \leq i \leq n$.

Then, by our definition of Σ-algebra, all subsort overloaded operators must agree on common data, so that we have,

$$f_{\mathcal{A}}^{s_1 \ldots s_n, s'}((t_1)_{\mathcal{A}}, \ldots, (t_n)_{\mathcal{A}}) = f_{\mathcal{A}}^{s'_1 \ldots s'_n, s''}((t_1)_{\mathcal{A}}, \ldots, (t_n)_{\mathcal{A}}).$$

Therefore, the definition does not depend on the choice of the subsort overloaded operator. As a consequence, the extension of $\_\mathcal{A}$ to the step $n+1$ is well-defined and, by construction, a Σ-homorphim, i.e., we have inductively proved the existence of the Σ-homomorphism $\_\mathcal{A}$. q.e.d.

More on Homomorphisms

Homomorphisms compose. That is, if $h : \mathcal{A} \longrightarrow \mathcal{B}$ and $g : \mathcal{B} \longrightarrow \mathcal{C}$ are Σ-homomorphisms, then $g \circ h = \{g_s \circ h_s\}_{s \in S}$ is a Σ-homomorphism $g \circ h : \mathcal{A} \longrightarrow \mathcal{C}$ (**Ex.**9.7).

Notation. The notation $g \circ h$ is the most common mathematical notation for function composition and is good to apply the composition to elements, since $g \circ h(x) = g(h(x))$ but has the unfortunate drawback of reversing the order of the arrows. Often we will use the alternative notation $h; g$ which is used for sequential composition in computer science and keeps the order of the arrows.

Identities are homomorphisms. That is, given a Σ-algebra $\mathcal{A} = (A, \_\mathcal{A})$, the family of identity functions $id_A = \{id_{A_s}\}$ is a Σ-homomorphim $id_A : \mathcal{A} \longrightarrow \mathcal{A}$.

More on Homomorphisms (II)

A Σ-homomorphim $h : \mathcal{A} \longrightarrow \mathcal{B}$ is called an isomorphim if there is another Σ-homomorphism $g : \mathcal{B} \longrightarrow \mathcal{A}$ such that $h; g = id_A$ and $g; h = id_B$. We then may use the notation $g = h^{-1}$ and $h = g^{-1}$.

We call a Σ-homomorphism $h : \mathcal{A} \longrightarrow \mathcal{B}$

- injective (resp. surjective) if for each sort $s \in S$ the function h_s is injective (resp. surjective)
- a monomorphism if for any pair of Σ-homomorphisms $g, q : \mathcal{C} \longrightarrow \mathcal{A}$, if $g; h = q; h$ then $g = q$
- an epimorphism if for any pair of Σ-homomorphisms $g, q : \mathcal{B} \longrightarrow \mathcal{C}$, if $h; g = h; q$ then $g = q$.

More on Homomorphisms (III)

For example, if $\mathbf{N}_{bin}$, resp. $\mathbf{N}_{dec}$, denote the natural numbers with 0, successor, and addition in binary, resp. decimal, representation, we have an obvious binary-to-decimal isomorphism $b2d : \mathbf{N}_{bin} \longrightarrow \mathbf{N}_{dec}$ preserving all operations, whose inverse is the decimal-to-binary isomorphism, $d2b : \mathbf{N}_{bin} \longrightarrow \mathbf{N}_{dec}$. Of course, $d2b; b2d = id_{\mathbf{N}_{dec}}$, and $b2d; d2b = id_{\mathbf{N}_{bin}}$.

For $\mathbf{N}_n$ the residue classes modulo n, the reminder function $\mathbf{N} \stackrel{rem_n}{\longrightarrow} \mathbf{N}_n$ is a surjective homomorphism for Σ containing, say, 0, 1, $+$, $\times$.

Similarly, for $\mathbf{Z}_{dec}$ the integers in decimal notation, the inclusion $j : \mathbf{N}_{dec} \hookrightarrow \mathbf{Z}_{dec}$ is an injective homomorphism preserving all shared operations: 0, 1, $+$, $\times$, etc.

Theorem: All Initial Algebras Are Isomorphic

Proof: Suppose $\mathcal{A}$ and $\mathcal{B}$ are Σ-algebras and both satisfy the initiality property of having a unique Σ-homomorphism to any other algebra. In particular, we have unique homomorphisms,

$$h : \mathcal{A} \longrightarrow \mathcal{B} \qquad g : \mathcal{B} \longrightarrow \mathcal{A}$$

and therefore a composed homomorphism

$$h; g : \mathcal{A} \longrightarrow \mathcal{B} \longrightarrow \mathcal{A}$$

but we also have the identity homomorphism id_A, which by uniqeness forces $h; g = id_A$. Interchanging the role of $\mathcal{A}$ and $\mathcal{B}$ we also get, $g; h = id_B$. q.e.d.

Evaluating Program Expressions

Q1: Can we model the evaluation of expressions in a programming language using initial algebras?

A1: We first of all need a signature Σ of operations.

For example, Σ could be a signature for integer operations, and/or Boolean operations, and/or real number operations (typically using a floating point representation).

Assume a programming language in which we only have integers and integer operations (note that we can encode true and false as, respectively, 0 and 1). In this case Σ can be unsorted and have two constants, 0 and 1, and three binary function symbols: $\_+\_$, $\_-\_$, and $\_*\_$.

Evaluating Program Expressions (II)

Q2: What else do we need?

A2: We need a set X of variables appearing on our expressions. This means that we need to extend Σ to $\Sigma(X)$, so that our program expressions will be terms $t \in T_{\Sigma(X)}$.

Q3: And what else do we need if we want to evaluate such expressions?

A3: We of course need a Σ-algebra in which they will be evaluated. For integers expressions this is the algebra $\mathcal{Z} = (\mathbf{Z}, \_{\mathcal{Z}})$ of the integers with $+, *, -, 0, 1$.

Evaluating Program Expressions (III)

Q4: And what else do we need?

A4: Since expression evaluation depends on the memory state, we need to model mathematically memory states.

Q5: And how can we model memory states?

A5: Assuming programs with just global variables, a memory state for arithmetic expressions is just a function $m : X \to \mathbf{Z}$. This is a special instance of the general notions of an assignment of values to variables in an algebra.

Assignments

Given variables in $X = \{X_s\}$ we will often be interested in assignments (also called valuations) of data elements in a given Σ-algebra $\mathcal{A} = (A, \_\mathcal{A})$ to those variables. Of course, if $x \in X_s$ then the value, say $a(x)$, assigned to x should be an element of A_s. That is the assignments should be well-sorted. All this can be made precise by defining an assignment as an S-indexed family of functions, $a = \{a_s : X_s \longrightarrow A_s\}_{s \in S}$, denoted $a : X \longrightarrow A$.

Often what we want to do with such assignments is to extend them from variables to terms on such variables in the obvious, homomorphic way.

Evaluating Program Expressions (VI)

Q6: Now that we have everything we need, how can evaluation of arithmetic expressions be precisely defined relative to a memory (state) $m : X \rightarrow \mathbf{Z}$?

A6: As a function $\_{(\mathcal{Z},m)} : T_{\Sigma(X)} \rightarrow \mathbf{Z}$ defined inductively by:

1. $x_{(\mathcal{Z},m)} = m(x)$ for $x \in X$
2. $0_{(\mathcal{Z},m)} = 0 \in \mathbf{Z}$, $1_{(\mathcal{Z},m)} = 1 \in \mathbf{Z}$
3. $f(t,t')_{(\mathcal{Z},m)} = f_{\mathcal{Z}}(t_{(\mathcal{Z},m)}, t'_{(\mathcal{Z},m)})$ for $f \in \{+,*,-\}$.

Evaluating Program Expressions (VII)

Q7: Conditions (2)–(3) show that $\_{(\mathcal{Z},m)}$ is a Σ-homomorphism. What about condition (1)?

A7: Condition (1) plus (2)–(3) show that it is a $\Sigma(X)$-homomorphism, when we extend the algebra $\mathcal{Z}$ of the integers with the additional constants X, where each $x \in X$ is interpreted in $\mathbf{Z}$ as $m(x)$. Denote this extended $\Sigma(X)$-algebra $(\mathcal{Z}, m)$. Then the evaluation of arithmetic expressions is the unique $\Sigma(X)$-homomorphism:

$$\_{(\mathcal{Z},m)} : \mathcal{T}_{\Sigma(X)} \to (\mathcal{Z}, m)$$

ensured by the initiality of $\mathcal{T}_{\Sigma(X)}$. So we have modeled expression evaluation as a homomorphism from $\mathcal{T}_{\Sigma(X)}$ to the $\Sigma(X)$-algebra $(\mathcal{Z}, m)$ extending $\mathcal{Z}$ with memory m.

Exercises

Ex.9.1. Show that a homomorphism is injective iff it is a monomorphism. Prove that every surjective homomorphism is an epimorphism. Construct an epimorphism that is not surjective.

Ex.9.2. Show that any many-sorted Σ-homomorphism that is surjective and injective is an isomorphism.

Construct an order-sorted homomorphism that is surjective and injective but is not an isomorphism. Give a sufficient condition on the poset $(S, \leq)$ (more general of course than being a discrete poset, since that is the many-sorted case) so that h is an isomorphism iff h is surjective and injective.

Exercises (II)

Ex.9.3. Prove that if an algebra $\mathcal{J}$ is isomorphic to an initial algebra $\mathcal{I}$, then $\mathcal{J}$ itself is initial.

Ex.9.4. Show that the natural numbers in Peano notation (zero and successor) and in base 2 are isomorphic Σ-algebras (both initial) for Σ the signature with one sort `Natural` and zero and successor operations.