

Semantic Algebras – Answers

1.

- a. *zero*
- b. (*false*, *true*)
- c. *two*
- d. *true*
- e. *three*
- f. *one*
- g. *zero*
- h. $\lambda m.$ *four*
- i. *one*

2.

- a. $[0, \{\}], [1, [\text{true}, \text{true}]], [1, [\text{true}, \text{false}]], [1, [\text{false}, \text{true}]], [1, [\text{false}, \text{false}]], [1, \perp]$
- b. $[0, \{\}], [1, [\text{true}, \text{true}]], [1, [\text{true}, \text{false}]], [1, [\text{false}, \text{true}]], [1, [\text{false}, \text{false}]], \perp$
- c. $[\{\}, \text{true}]$ or $[\{\}, \text{false}]$ or $[\{\}, \perp]$
- d. $([\{\}, \text{true}]$ or $[\{\}, \text{false}])$ and \perp

3.

- a. *one cons nil*
- b. *one cons nil*
- c. no simplification
- d. \perp

4. The original semantic algebra was

Domain $Array = Nat \rightarrow A$

Operations

$newarray : Array$

$newarray = \lambda n.error$

$access : Nat \rightarrow Array \rightarrow A$

$access = \lambda n.\lambda r.r(n)$

$update : Nat \rightarrow A \rightarrow Array \rightarrow Array$

$update = \lambda n.\lambda v.\lambda r.[n \mapsto v]r$

We need to change it to add the upper and lower bounds. The simplest way is to alter the domain to include the bounds with the function, and then test the upper and lower bounds in *access* and *update*. The domain now contains a triple:

Domain $Array = (Nat \rightarrow A) \times Nat \times Nat + Error$, where A is any domain with the *error* element, and *Error* is a *Unit* domain used to signal an error during updating. It contains only one value: *errorarray*.

newarray becomes:

newarray: $\text{Nat} \times \text{Nat} \rightarrow \text{Array}$
newarray = $\lambda l. \lambda u. [\lambda n. \text{error}, l, u]$

In other words, creating an array needs two numbers, upper and lower bounds. E.g.

newarray zero ten

returns a triple containing *zero* and *ten*:

$a_{\text{new}} = [\lambda n. \text{error}, \text{zero}, \text{ten}]$

Access and update are then:

access : $\text{Nat} \times \text{Array} \rightarrow A$
access = $\lambda n. \lambda [r, l, u]. n \text{ greater } u \rightarrow \text{error}$
 $\square n \text{ less } l \rightarrow \text{error}$
 $\square r(n)$

update : $\text{Nat} \times \text{Nat} \times \text{Array} \rightarrow \text{Array}$
update = $\lambda n. \lambda v. \lambda [r, l, u]. n \text{ greater } u \rightarrow \text{errorarray}$
 $\square n \text{ less } l \rightarrow \text{errorarray}$
 $\square [[n \mapsto v] r, l, u]$

e.g. *update three twelve* $[\lambda n. \text{error}, \text{zero}, \text{ten}] = [[\text{three} \mapsto \text{twelve}](\lambda n. \text{error}), \text{zero}, \text{ten}]$

and *update eleven four* $[[\text{three} \mapsto \text{twelve}](\lambda n. \text{error}), \text{zero}, \text{ten}] = \text{errorarray}$

Then *access three* $[[\text{three} \mapsto \text{twelve}](\lambda n. \text{error}), \text{zero}, \text{ten}] = \text{twelve}$

and *access eleven* $[[\text{three} \mapsto \text{twelve}](\lambda n. \text{error}), \text{zero}, \text{ten}] = \text{error}$