ELEMENTS OF PROGRAMMING SOLUTIONS

ASEF AHMED

1. Foundations

1.1. Categories of Ideas: Entity, Species, Genus.

1.2. Values.

Lemma 1.1. *If a value type is uniquely represented, equality implies representational equality.*

Solution. Suppose a value type $\mathtt{T}$ is uniquely represented. Denote by $\mathtt{v}, \mathtt{v}' : \mathtt{T}$ as equal values of $\mathtt{T}$. By unique representation, $\mathtt{v}, \mathtt{v}'$ each correspond uniquely to the abstract entities $\mathtt{E}, \mathtt{E}'$, and by equality of values, these entities must also be equal. Hence the data $\mathtt{D}, \mathtt{D}'$ for $\mathtt{v}, \mathtt{v}'$ are identical, and so $\mathtt{v}, \mathtt{v}'$ are representationally equal. □

Lemma 1.2. *If a value type is not ambiguous, representational equality implies equality.*

Solution. Suppose a value type $\mathtt{T}$ is not ambiguous. Denote by $\mathtt{v}, \mathtt{v}' : \mathtt{T}$ as representationally equal values of $\mathtt{T}$. As $\mathtt{T}$ is not ambiguous, $\mathtt{v}, \mathtt{v}'$ must each have at most one interpretation, and by representational equality, the data $\mathtt{D}, \mathtt{D}'$ for the values are identical. Hence the values $\mathtt{v}, \mathtt{v}'$ must represent the same abstract entity $\mathtt{E}$, and so they are equal. □

1.3. Objects.

1.4. Procedures.

1.5. Regular Types.

Lemma 1.3. *A well-formed object is partially formed.*

Solution. Suppose $\mathtt{a}$ is an object that is well-formed. Let $\mathtt{S_a}$ be the state of $\mathtt{a}$, which by definition is a value $\mathtt{v} : \mathtt{T}$ of some value type $\mathtt{T}$. By well-formedness of $\mathtt{a}$, $\mathtt{S}$ is also well-formed as a value, i.e. WLOG, we may assume that $\mathtt{T}$ is `double` as an object type for $\mathtt{a}$. Let $\mathtt{b}$ be another object of type `double`. Certainly $\mathtt{a}$ may be assigned to $\mathtt{b}$ without modifying the state $\mathtt{S_b}$ of $\mathtt{b}$, and $\mathtt{a}$ may be destructed as well. Therefore $\mathtt{a}$ is partially formed. □

1.6. Regular Procedures.

Exercise 1.1. *Extend the notion of regularity to input/output objects of a procedure, that is, to objects that are modified as well as read.*

Solution. A procedure is *regular* if and only if the input objects $\mathtt{a}_0, \dots, \mathtt{a_r}$, when replaced by equal objects $\mathtt{b}_0, \dots, \mathtt{b_r}$, i.e. for each $i \in \{0, \dots, r\}$, the states $\mathtt{S_{a_i}} = \mathtt{S_{b_i}}$ are equivalent for the objects $\mathtt{a_i}, \mathtt{b_i}$ of type $\mathtt{T_i}$, yield equivalent output objects $\mathtt{c_i} = \mathtt{d_i}$ (where equality of objects means equivalence of the corresponding states). There is a natural equivalence for input/output objects in the following way: an input/output object $\mathtt{s}$ is equivalent to $\mathtt{t}$, i.e. $\mathtt{s} \equiv \mathtt{t}$, if there is a regular procedure $\mathtt{F}$ for which $\mathtt{t}$ is the output object of the input $\mathtt{s}$ under $\mathtt{F}$, or symmetrically if $\mathtt{s}$ is the output object of the input $\mathtt{t}$ under $\mathtt{F}$. □